R1000 Systems

For the R1000 systems there were 4 "Series" created; 100, 200, 300 and 400. With the progression to each new system the changes were mainly in the IO area, and the packaging changes enabled by these changes. The packaging changes, smaller physical frame size, was mostly a function of Disk /Tape Drive sizes.

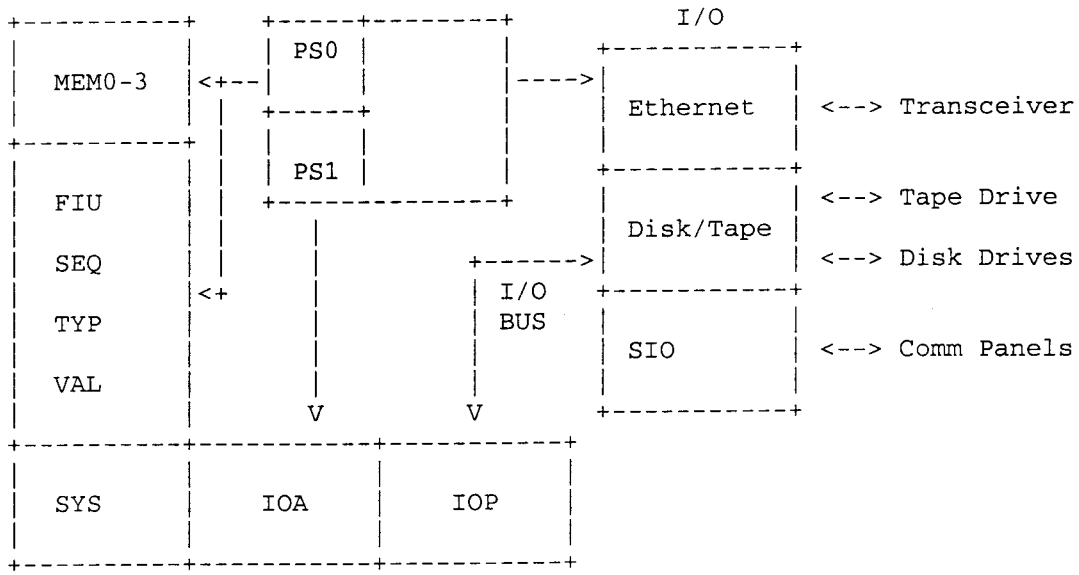
An R1000 system can be broken up into several major functional blocks. The R1000 module is composed of several boards which are the CPU and Memory. The R1000 does not directly control its peripherals, rather issues commands to the I/O Processor module which directly communicates with the peripheral controller boards located in the I/O cardcage. The path which the R1000 takes to communicate with the IOP is through the I/O Adapter module, which functionally has control over all modules in the system, and is the first active module in the boot process, providing the physical interface to the operations console and diagnostic modem, running self tests on all modules (IOA, IOP, R1000 boards, etc...) and controlling the power to the R1000 board.

The IOP and IOA reside on separate boards on the series 100. On the series 200 and later, the IOP, IOA, and SYSBUS modules all reside on a single board called the I/O Controller (IOC). With the 400 Series the system moved from the use of third-party peripheral controller boards, to implementing these functions on our own board. The new board being the RESHA.

Across the Series, the CPU itself has not really changed. The CPU is based on TTL logic. With the creation of the Series 400, the edge connectors were changed.

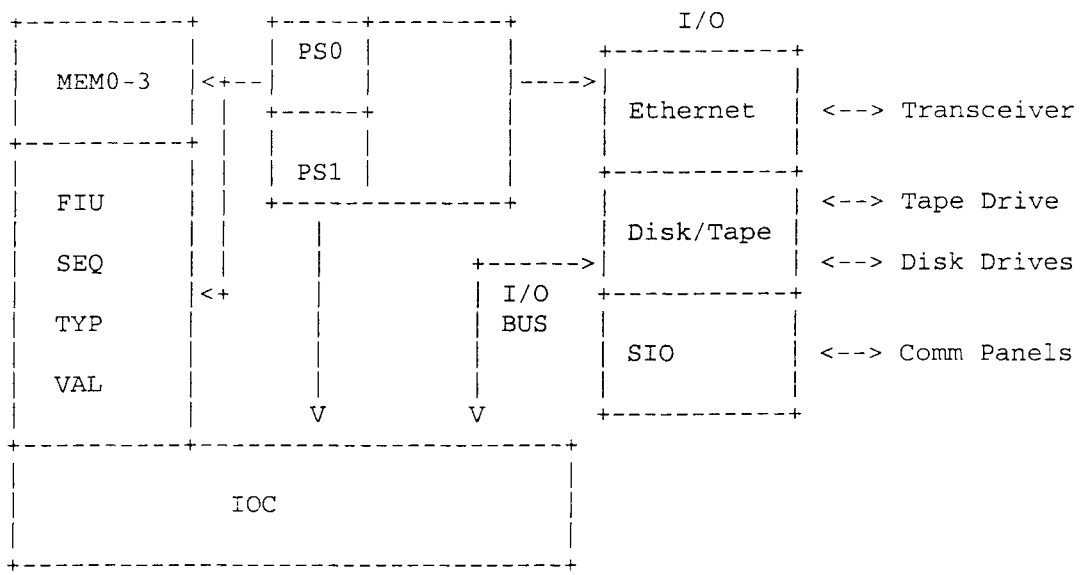
Following are some basic diagrams of the different Systems. More detail information can be found for the 200 through 400 in the System Manager's Guide.

SERIES 100


I/O
PS0
MEM0-3
Ethernet
<--> Transceiver
PS1
FIU
Tape Drive
Disk/Tape
SEQ
Disk Drives
I/O
BUS
TYP
SIO
Comm Panels
VAL
SYS
IOA
IOP

SERIES 200 /300

Main change was to consolidate the SYS,IOA and IOP into the new IOC board.

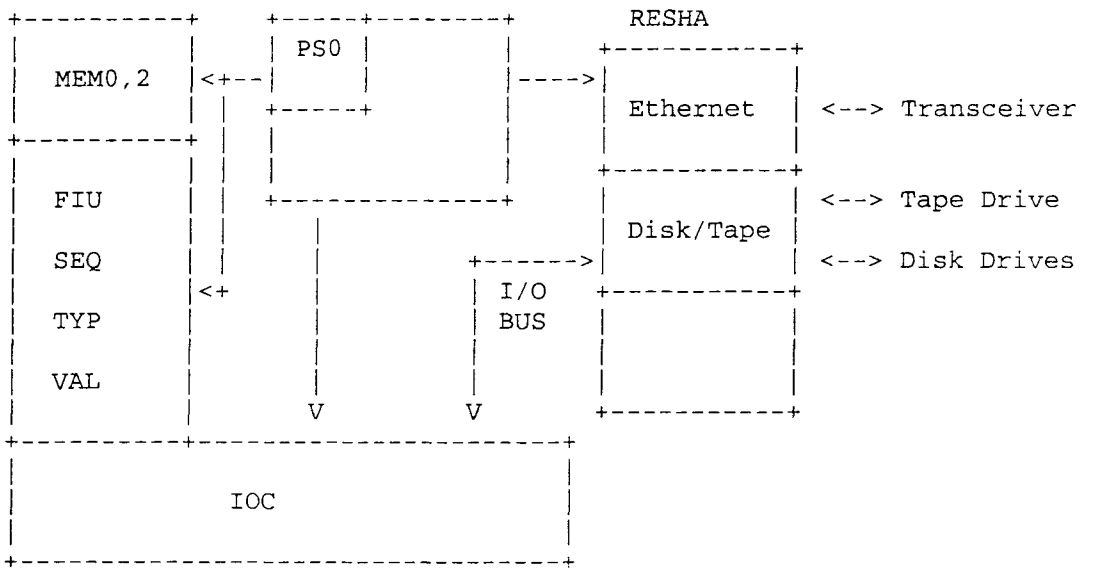


SERIES 300

Introduced 32 meg memory boards, allowing systems to have up to 64 meg of memory. Moved to using 8mm tape drive as standard tape unit. Disk drive in separate frame, and system could be setup to either use local disk, or access drives on another server over a "coprocessor" link.

SERIES 400

Introduced Resha Board that implement all IO. System only supported 32 meg memory boards. A coprocessor version was initially implemented by due to lack of interest and complexities of implementation it was dropped. Moved to SCSI disk drives.



Series 400 Boards Set

The R1000 processor is composed of a CPU board set and memory. The boards are housed in the R1000 board section and have both a backplane and a foreplane. Access to the boards requires removal of the foreplane. The location of the boards, when viewing the system from the front, and starting from the left to right are:

CPU Board Set

Slot	Board	Description
1	Mem2	Memory 2 (32 meg)
2	Mem0	Memory 0 (32 meg)
3	FIU	Field Isolation Unit
4	SEQ	Sequencer
5	TYP	Type
6	VAL	Value
7	IOC	I/O Controller

To the right of the CPU boards is the RESHA. This card connects only through the backplane and does not share the foreplane. The RESHA implements the connection and communication to peripherals. The front edge of the RESHA is the Input/Output Panel that contains:

- Two RS232 ports (Comm and operator's console)
- network ports
- Other assorted devices interfaces, switches and status LEDs.

RESHA (Rational Ethernet SCSI Host Adapter)

All I/O in one. Contains SCSI controller for Disk and Tape drive, Control Panel logic, VMEGEN board for CMC Ethernet Controller, PCM functions, Diagnostic Modem, "White Button", EEPROM that contains Bootstrap Software (Disk, Tape)and Self-test.

IOC (Input/Output Controller)

I/O Processor (68020)
I/O Processor Memory
Memory Error Correction
Microaddress Trace RAM
Bus Control Logic
IO bus Generation
Operator Console Interface
EEPROM – containing
  IOC selftest
  Bootstrap manager (Boot/Crash options and IOP configuration)
  NOVRAM that holds IOC serial number, cluster ID, etc...
  Battery and Clock/Calendar Chip

VAL (Value)
Responsible for VALUE calculations
Contains 16 x 16 multiplier, 64-bit Arithmetic/Logical Unit
1K x 64 Bit Register File (Includes control Stack accelerator)

TYP (Type)
Contains Type checking hardware
Memory address register
64-bit Arithmetic/Logical Unit
1K x 64 Bit Register File (Includes control Stack accelerator)

SEQ (Sequencer)
Macro-instruction processing
Micro-instruction sequencing

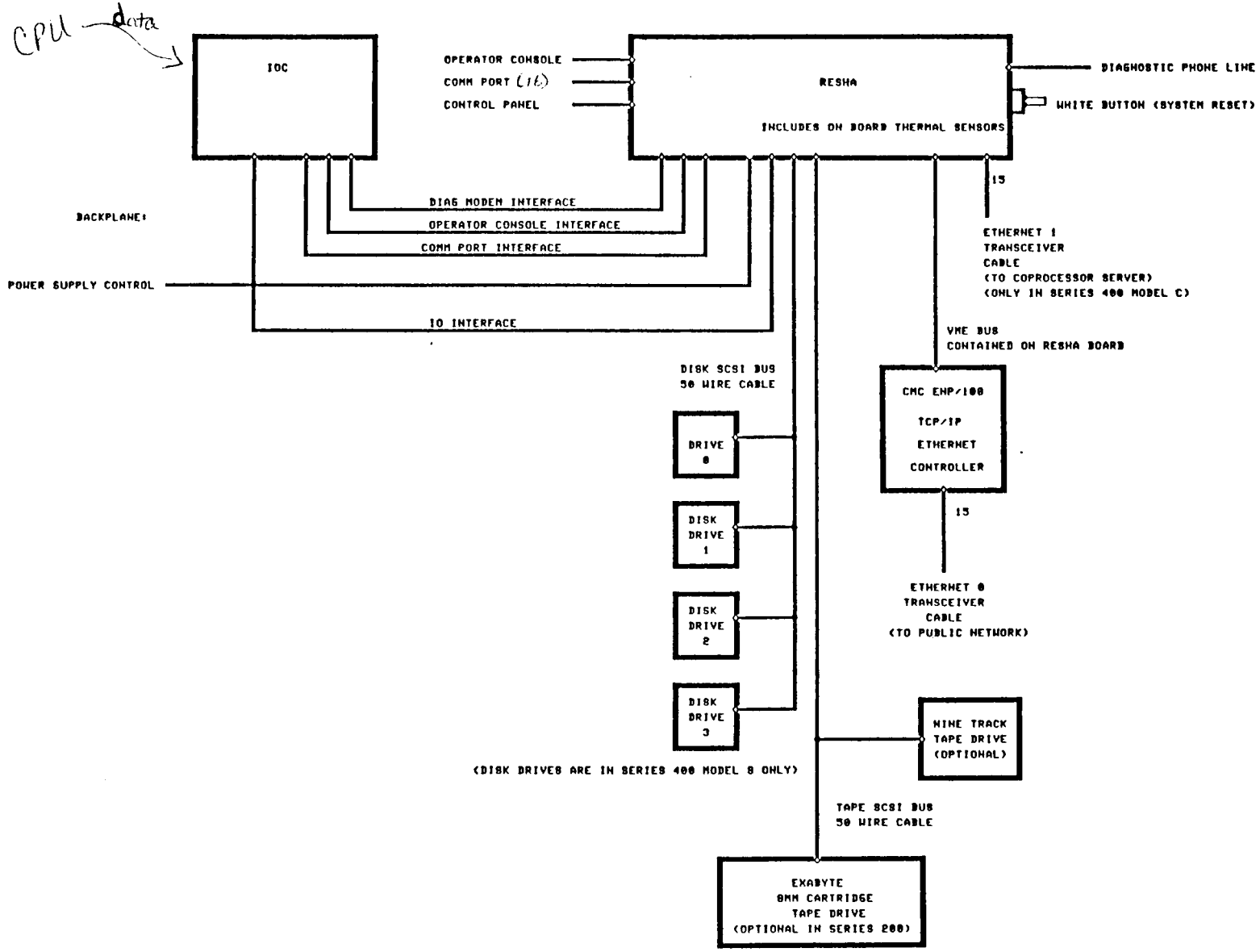
FIU (Field Isolation Unit)
Contains Control Stack Accelerator logic
Memory Control logic
64 into 128 Bit Field Insertion
128 to 64 Bit Field Extraction

MEM32 (32 MB Memory)
Main memory for system, one board required for a total of 32 MB, organized in words of 137 bits (128 data bits and 9 ECC bits)
Memory is divided into 1024 byte pages
Memory boards response directly to Virtual Addresses

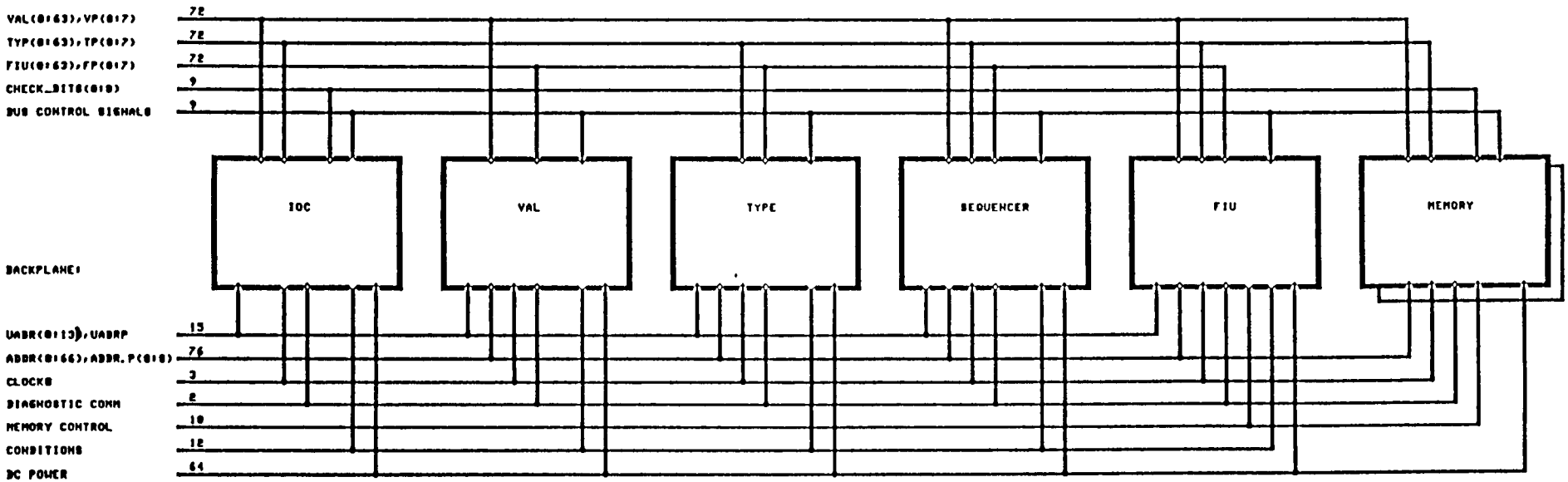
Tape Drive
The series 400 makes use of an Exabyte EXB-8200 8mm tape drive. The system is setup to house on one of these drives.

Disk Drives
The series 400 makes use of 5-1/4 Inch SCSI disk drives. The system driver software is limited to support of only the Fujitsu M2266 drive. The drives are mounted on a special series 400 specific carrier that allows for them to be easily installed and removed from the systems disk backplane. The system will support up to four drives on the backplane. The position of the drive on the backplane determines unit number.

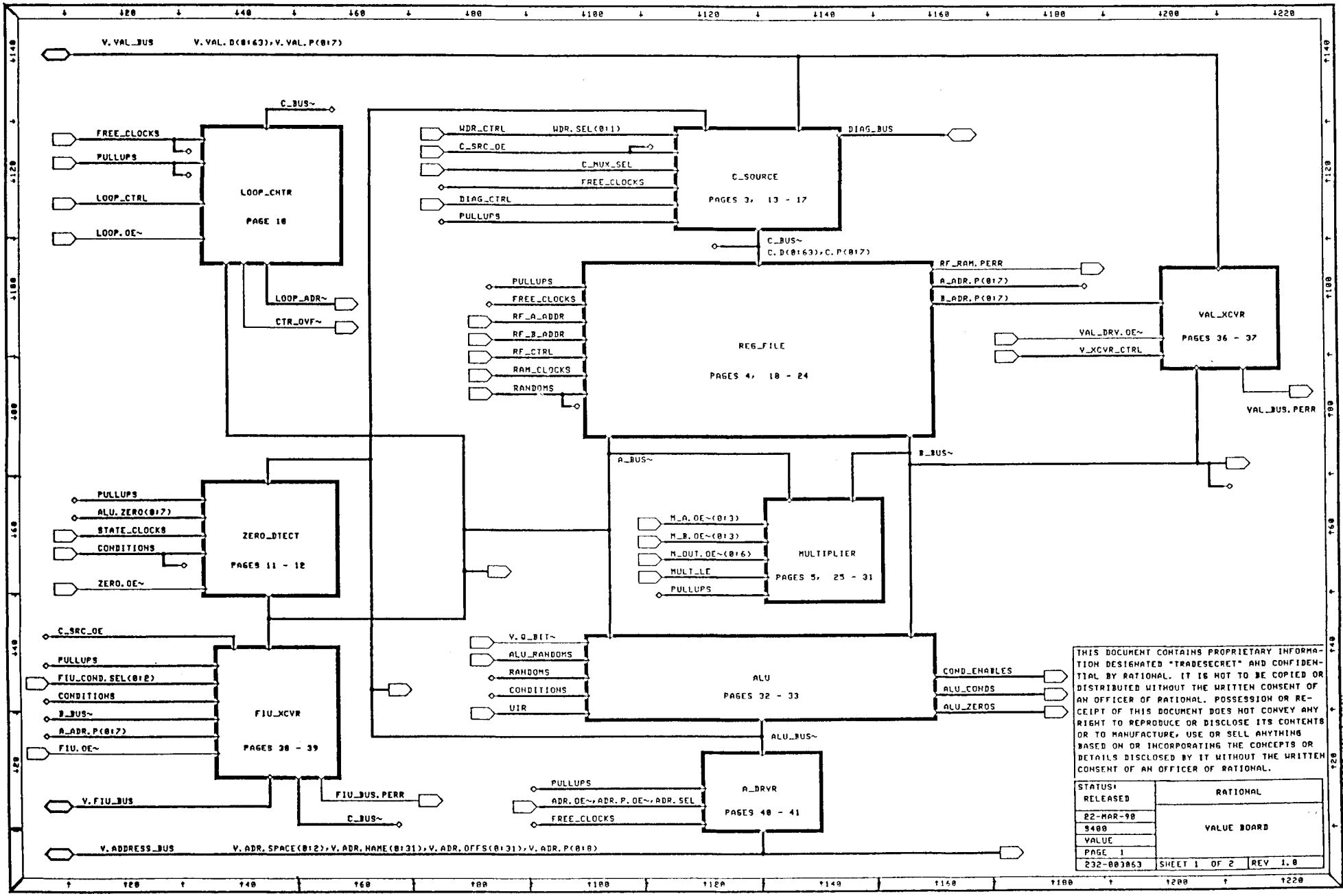
DC Power
One Power Supply
PS0 for CPU and Peripherals +5VDC, +12VDC, -12VDC



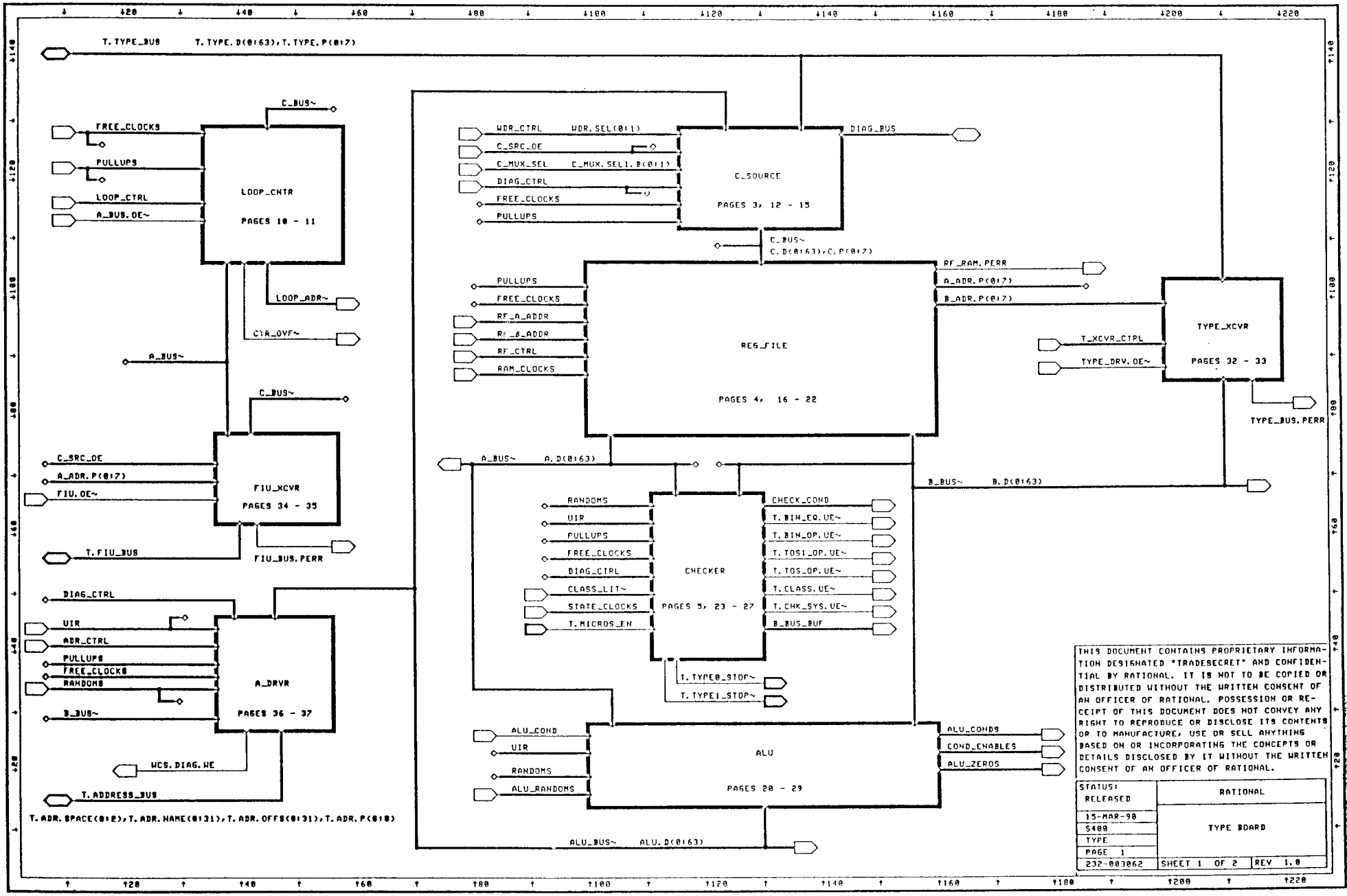
STATUS:	RATIONAL
17-SEP-90	R1000 IO DIAGRAM
R1000	SERIES 400
	SHEET 1 OF 1 | REV 0.



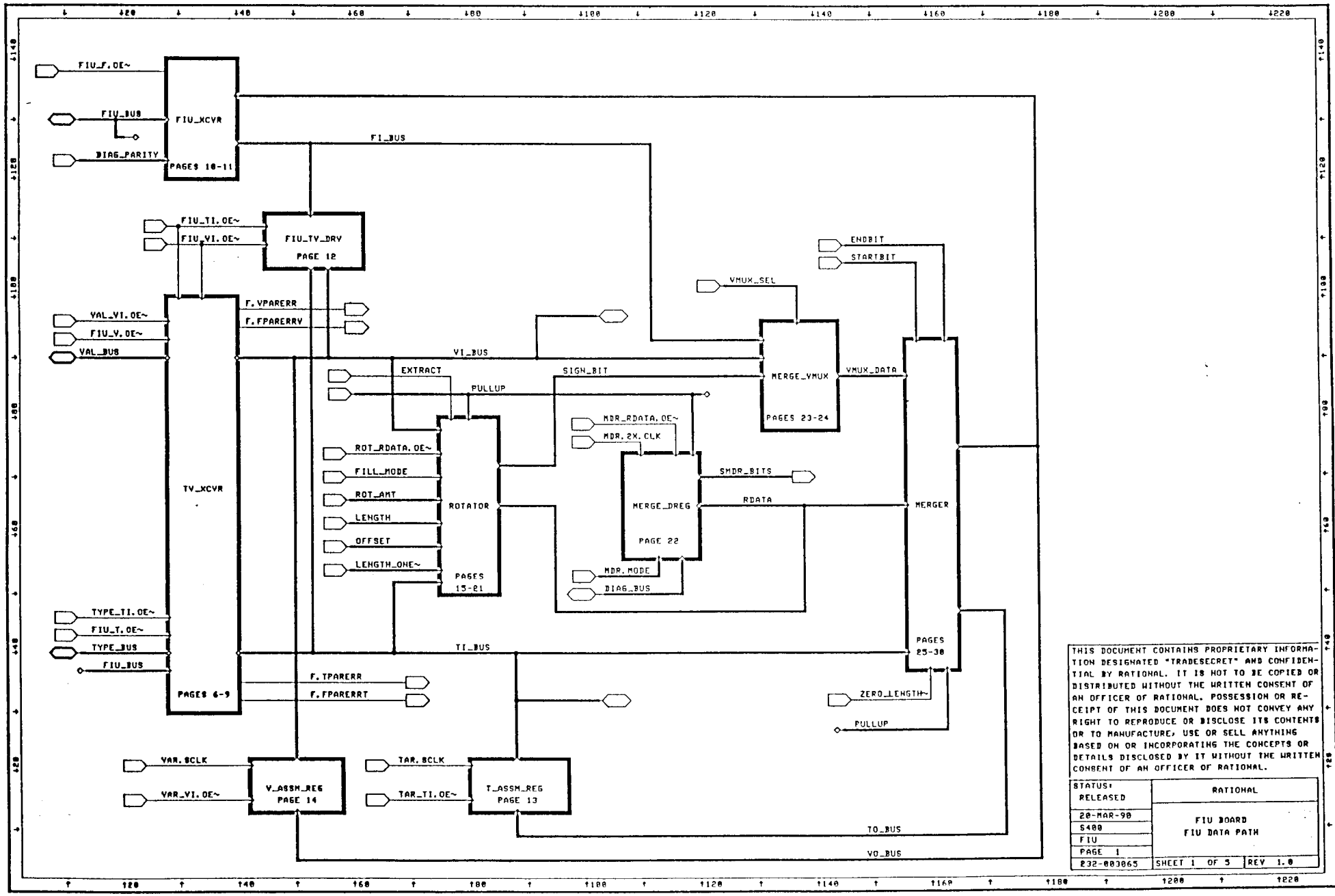
STATUS:	RATIONAL
17-SEP-90	R1000 CPU
R1000	INTERCONNECT DIAGR
	SHEET 1 OF 1 | REV



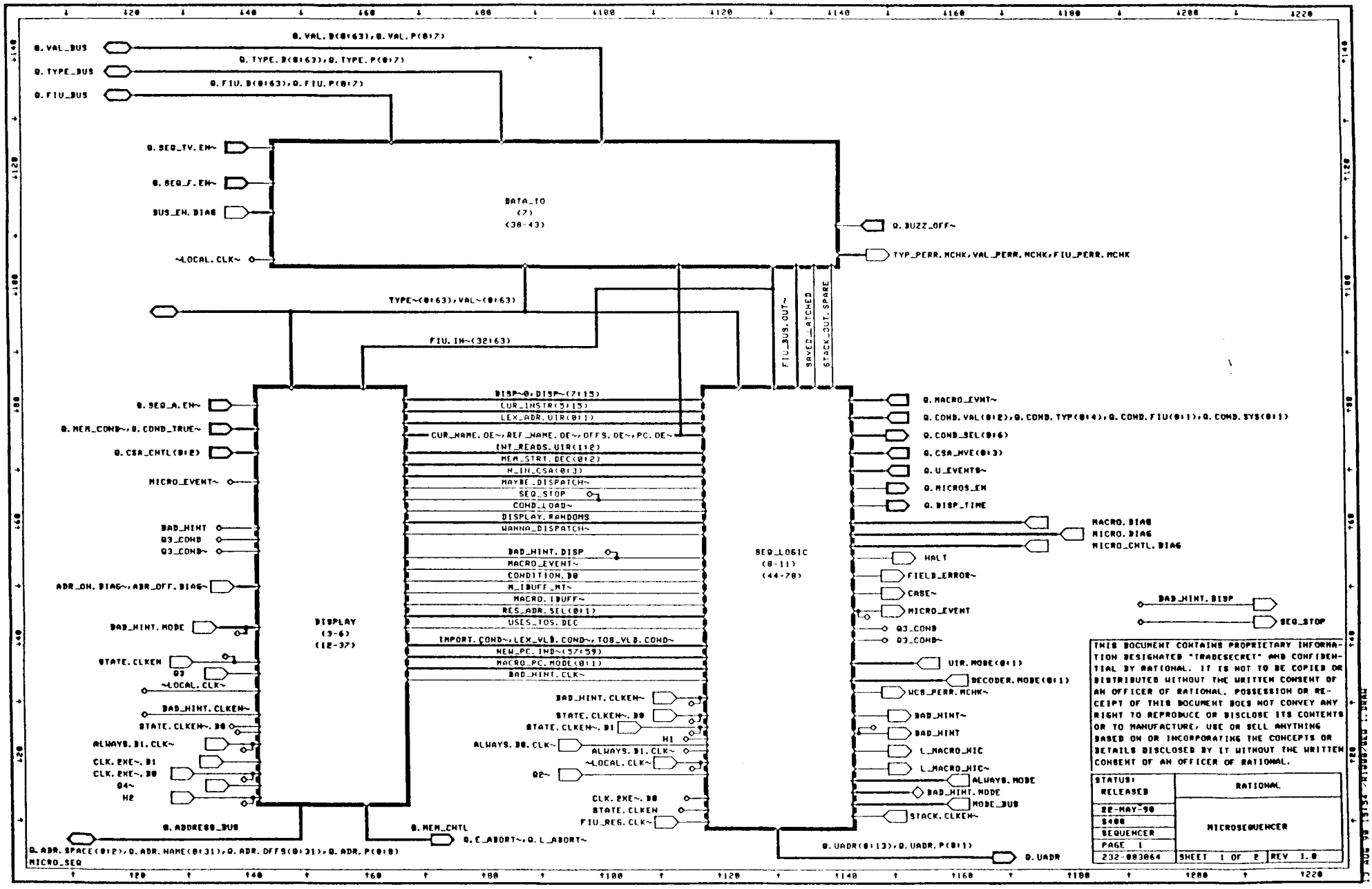
STATUS: RELEASED	RATIONAL
22-MAR-90	VALUE BOARD
9400	
VALUE	
PAGE 1	
232-003063	SHEET 1 OF 2 REV 1.0

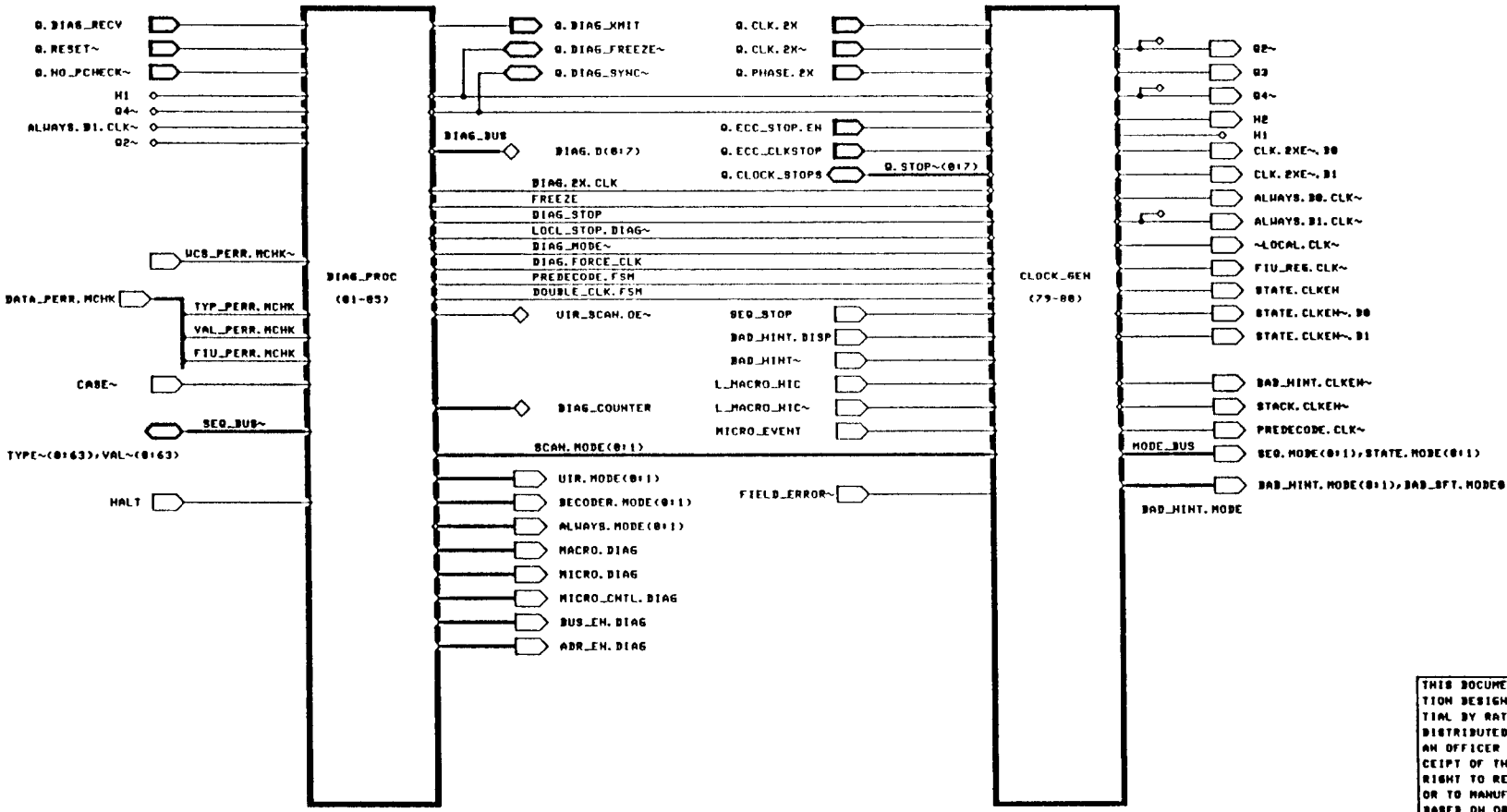


STATUS: RELEASED	RATIONAL	
15-MAR-90	TYPE BOARD	
S400		
TYPE		
PAGE 1		
232-003062	SHEET 1 OF 2	REV 1.0



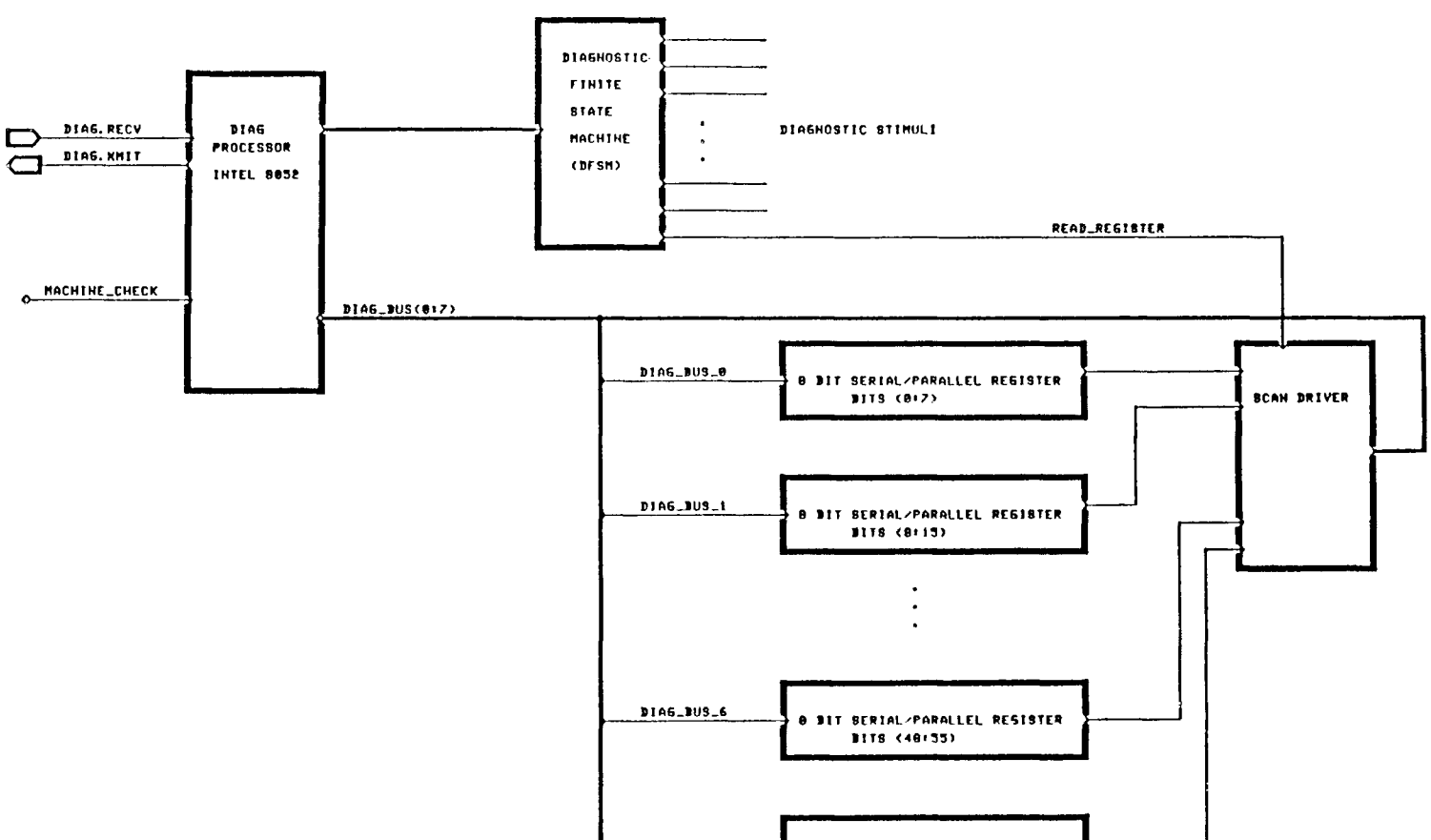
STATUS: RELEASED	RATIONAL
20-MAR-90	FIU BOARD FIU DATA PATH
S400	
FIU	
PAGE 1	
232-003065	SHEET 1 OF 5 REV 1.0


Q.VAL_BUS
Q.VAL.D(0:63),Q.VAL.P(0:7)
Q.TYPE_BUS
Q.TYPE.D(0:63),Q.TYPE.P(0:7)
Q.FIU_BUS
Q.FIU.D(0:63),Q.FIU.P(0:7)
Q.SEQ_TV.EN~
Q.SEQ_F.EN~
BUS_EN.DIAG
~LOCAL.CLK~
DATA_IO
(7)
(38-43)
Q.BUZZ_OFF~
TYP_PERR.MCHK,VAL_PERR.MCHK,FIU_PERR.MCHK
TYPE~(0:63),VAL~(0:63)
FIU.IN~(32:63)
FIU_BUS.OUT~
SAVED_LATCHED
STACK_OUT.SPARE
Q.SEQ_A.EN~
Q.MEM_COND~,Q.COND_TRUE~
Q.CSA_CNTL(0:2)
MICRO_EVENT~
BAD_HINT
Q3_COND
Q3_COND~
ADR_OH.DIAG~,ADR_OFF.DIAG~
BAD_HINT.MODE
STATE.CLKEN
Q3
~LOCAL.CLK~
BAD_HINT.CLKEN~
STATE.CLKEN~.B0
ALWAYS.B1.CLK~
CLK.EXE~.B1
CLK.EXE~.B0
Q4~
H2
DISPLAY
(3-6)
(12-37)
DISP~0,DISP~(7:15)
CUR_INSTR(5:15)
LEX_ADR.UIR(0:1)
CUR_NAME.OE~,REF_NAME.OE~,OFFS.OE~,PC.OE~
INT_READS.UIR(1:2)
MEM_STRT.DEC(0:2)
M_IN_CSA(0:3)
MAYBE_DISPATCH~
SEQ_STOP
COND_LOAD~
DISPLAY.RANDOMS
WANNA_DISPATCH~
BAD_HINT.DISP
MACRO_EVENT~
CONDITION.B0
M_IBUFF_MT~
MACRO.IBUFF~
RES_ADR.SEL(0:1)
USES_TOS.DEC
IMPORT.COND~,LEX_VLD.COND~,TOS_VLD.COND~
NEW_PC.IND~(57:59)
MACRO_PC.MODE(0:1)
BAD_HINT.CLK~
SEQ_LOGIC
(8-11)
(44-78)
BAD_HINT.CLKEN~
STATE.CLKEN~.B0
STATE.CLKEN~.B1
H1
ALWAYS.B0.CLK~
ALWAYS.B1.CLK~
~LOCAL.CLK~
Q2~
CLK.EXE~.B0
STATE.CLKEN
FIU_REG.CLK~
Q.MACRO_EVNT~
Q.COND.VAL(0:2),Q.COND.TYP(0:4),Q.COND.FIU(0:1),Q.COND.SYS(0:1)
Q.COND_SEL(0:6)
Q.CSA_MVE(0:3)
Q.U_EVENTS~
Q.MICROS_EN
Q.DISP_TIME
MACRO.DIAG
MICRO.DIAG
MICRO_CNTL.DIAG
HALT
FIELD_ERROR~
CASE~
MICRO_EVENT
Q3_COND
Q3_COND~
UIR.MODE(0:1)
DECODER.MODE(0:1)
WCS_PERR.MCHK~
BAD_HINT~
BAD_HINT
L_MACRO_HIC
L_MACRO_HIC~
ALWAYS.MODE
BAD_HINT.MODE
MODE_BUS
STACK.CLKEN~
BAD_HINT.DISP
SEQ_STOP
Q.ADDRESS_BUS
Q.ADR.SPACE(0:2),Q.ADR.NAME(0:31),Q.ADR.OFFS(0:31),Q.ADR.P(0:8)
MICRO_SEQ
Q.MEM_CNTL
Q.E_ABORT~,Q.L_ABORT~
Q.UADR(0:13),Q.UADR.P(0:1)
Q.UADR
THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION DESIGNATED "TRADESECRET" AND CONFIDENTIAL BY RATIONAL. IT IS NOT TO BE COPIED OR DISTRIBUTED WITHOUT THE WRITTEN CONSENT OF AN OFFICER OF RATIONAL. POSSESSION OR RECEIPT OF THIS DOCUMENT DOES NOT CONVEY ANY RIGHT TO REPRODUCE OR DISCLOSE ITS CONTENTS OR TO MANUFACTURE, USE OR SELL ANYTHING BASED ON OR INCORPORATING THE CONCEPTS OR DETAILS DISCLOSED BY IT WITHOUT THE WRITTEN CONSENT OF AN OFFICER OF RATIONAL.
STATUS: RELEASED
22-MAY-90
S400
SEQUENCER
PAGE 1
232-003064
RATIONAL
MICROSEQUENCER
SHEET 1 OF 2
REV 1.0



STATUS: RELEASED	RATIONAL
22-MAY-90	CLOCKS AND DIAGNOSTICS
R400	
SEQUENCER	
PAGE 2	
232-003864	SHEET 2 OF 2 REV 1.0

CPU - FRUs



STATUS:	RATIONAL
17-SEP-90	R1000 DIAGNOSTIC SYST
	PER BOARD HARDWARE
R1000	
	SHEET 1 OF 1 REV 0.0

R1000 Microcode

The R1000 implements it instruction set and system control via Microcode.

What is Microcode

- small, subroutine type program, controlling system hardware
- implements Instruction Set and System Control
- called – Ucode, Firmware, Microcode

What are characteristics on an R1000

- Writable Control Store
  * allows easy update/bug fixes
  * more than one version of microcode (diagnostic vs production)
  * patches can be made

- Ada Specific
  * checking built into hardware and controlled by microcode
  * Ada tasking, and package elaboration built into microcode

- Very Complex Instruction Set
  * Complex Type and Variable Declaration and execute implemented in microcode
  * System Scheduling done at microcode level

The two classes of Microcode

With the R1000 Writable Control Store it is possible to select various versions of microcode to be loaded into the machine. The R1000 has basically two forms of microcode; Diagnostic microcode, which is used to verify that the hardware is operating correctly, and Production microcode, which is the microcode running while the machine is in normal use. The production microcode defines the instruction set and operating characteristics of the R1000.

Diagnostic Microcode

- "Climbs" onto the Hardware, first testing basic operation, and then moving on to test more and more complex functions
- Runs "at speed", which is to say it executes at the normal clock rate, rather than the single step approach of the FRUs.
- The Microdiagnostic is broken into sections that focus on testing a specific board. On a failure the halting address will give a good indication of the board that is at fault. By using the EXPMON to check register results and trace information it is possible to get a further understanding of the failure, possibly down to the failing component.

Production Microcode

- Defines the R1000 Instruction Set
- Specifically implements support for the ADA language
- Manages ADA tasking and Package/Procedure elaboration and execution
- Manages system resources
- Low level job scheduling
- Loaded from the DFS

Micrcode Example

```
0FCC :   SELECT_CONDITION (TYP (TRUE)),
      : ;
0FCD :   DISABLE (MICRO_EVENTS),
      :   TYP { RESULT := PASS_HIGH_A (16#36C936C936C936C9#) },
      :   TYP { RESULT := PASS_A(16#36C936C936C936C9#) ELSE
      :            INC_A (16#36C936C936C936C9#) },
      : ;
0FCE :   TYP { BAD_BITS := RESULT XOR 16#36C936C936C936C9# },
      :   IF TYP (RESULT /= 16#36C936C936C936C9#) THEN
      :            RARELY CALL SPLIT_ALU_ERROR,
      : ;
0FCF :   SELECT_CONDITION (TYP (FALSE)),
      : ;
0FD0 :   DISABLE (MICRO_EVENTS),
      :   TYP { RESULT := PASS_HIGH_A (16#36C936C936C936C9#) },
      :   TYP { RESULT := PASS_A(16#36C936C936C936C9#) ELSE
      :            INC_A (16#36C936C936C936C9#) },
      : ;
0FD1 :   TYP { BAD_BITS := RESULT XOR 16#36C936C936C936CA# },
      :   IF TYP (RESULT /= 16#36C936C936C936CA#) THEN
      :            RARELY CALL SPLIT_ALU_ERROR,
      : ;
0FD2 :   SELECT_CONDITION (TYP (TRUE)),
      : ;
0FD3 :   DISABLE (MICRO_EVENTS),
      :   TYP { RESULT := PASS_HIGH_A (16#36C936C936C936C9#) },
      :   TYP { RESULT := INC_A (16#36C936C936C936C9#) ELSE
      :            PASS_A(16#36C936C936C936C9#) },
      : ;
0FD4 :   TYP { BAD_BITS := RESULT XOR 16#36C936C936C936CA# },
      :   IF TYP (RESULT /= 16#36C936C936C936CA#) THEN
      :            RARELY CALL SPLIT_ALU_ERROR,
 .
 .
 .

116F : SPLIT_ALU_ERROR:
     :   DISABLE(MICRO_EVENTS),
     :   HALT,
     : ;
1170 :   RETURN,
     :;
```

Example code from Discrete_Execute

```
    PLUS_OP                     : dispatch target;  -- 1
    MINUS_OP                    : dispatch target;  -- 1


--TITLE DISCRETE_EXECUTE  -  PLUS_OP

OPCODE (NAME         => "EXECUTE,DISCRETE,PLUS",
        LABEL        => PLUS_OP,
        NEEDS_VALID => 2,
        NEEDS_FREE   => 0);

PLUS_OP:
    CHECK_CLASS    ([TOS-1] = [TOS] = OF_KIND.DISCRETE_VAR),
    TYP { [TOS-1] := LITERAL_DISCRETE },
    VAL { [TOS-1] := [TOS-1] + [TOS]  },
    POP_CONTROL_STACK,
    IF NOT VAL(OVERFLOW) THEN USUALLY DISPATCH,
;
    DISABLE (MICRO_EVENTS),
    NOT_RESTARTABLE,
    CALL RAISE.OVERFLOW_ERROR
;
--TITLE DISCRETE_EXECUTE  -  MINUS_OP

OPCODE (NAME         => "EXECUTE,DISCRETE,MINUS",
        LABEL        => MINUS_OP,
        NEEDS_VALID => 2,
        NEEDS_FREE   => 0);

MINUS_OP:
    CHECK_CLASS    ([TOS-1] = [TOS] = OF_KIND.DISCRETE_VAR),
    TYP { [TOS-1] := LITERAL_DISCRETE },
    VAL { [TOS-1] := [TOS-1] - [TOS]  },
    POP_CONTROL_STACK,
    IF NOT VAL(OVERFLOW) THEN USUALLY DISPATCH,
;
    DISABLE (MICRO_EVENTS),
    NOT_RESTARTABLE,
    CALL RAISE.OVERFLOW_ERROR
;
```

```
      : package DIAG_REGS is
      :
      :     PASS_COUNTER  : VGP(0);
      :     RESULT        : GP(1);
      :     BAD_BITS      : GP(2);
      :     PATTERN_1     : GP(3);
      :     PATTERN_2     : GP(4);
      :     VGP_5         : VGP(5);
      :     TGP_5         : TGP(5);
      :     GP_5          : GP(5);
      :     VGP_6         : VGP(6);
      :     TGP_6         : TGP(6);
      :     GP_6          : GP(6);
      :     VGP_7         : VGP(7);
      :     TGP_7         : TGP(7);
      :     GP_7          : GP(7);
      :     VGP_8         : VGP(8);
      :     TGP_8         : TGP(8);
      :     GP_8          : GP(8);
      :     VGP_9         : VGP(9);
      :     TGP_9         : TGP(9);
      :     GP_9          : GP(9);
      :     VGP_10        : VGP(10);
      :     TGP_10        : TGP(10);
      :     GP_10         : GP(10);
      :     VGP_11        : VGP(11);
      :     TGP_11        : TGP(11);
      :     GP_11         : GP(11);
      :     VGP_12        : VGP(12);
      :     TGP_12        : TGP(12);
      :     GP_12         : GP(12);
      :     VGP_13        : VGP(13);
      :     TGP_13        : TGP(13);
      :     GP_13         : GP(13);
      :     VGP_14        : VGP(14);
      :     TGP_14        : TGP(14);
      :     GP_14         : GP(14);
      :     VGP_15        : VGP(15);
      :     TGP_15        : TGP(15);
      :     GP_15         : GP(15);
      : end DIAG_REGS;
      :
0100  : DIAGNOSTIC_START:
      :     VAL { PASS_COUNTER := ZEROS },
      :     GOTO DIAGNOSTIC_DRIVER.START,
      : ;
      :
0101  : END_DIAGNOSTIC_PASS:
      :     VAL { ALU_BUS := PASS_A (PASS_COUNTER) },
      :     IF VAL (ALU_BUS /= 0) THEN USUALLY GOTO NOT_FIRST_PASS,
      : ;
0102  : COMPLETED_FULL_TEST:
      :     -- Halt at end of first pass --
      :     HALT,
      : ;
      :
0103  : NOT_FIRST_PASS:
      :     VAL { PASS_COUNTER := INC_A (PASS_COUNTER) },
      :     GOTO DIAGNOSTIC_DRIVER.START,
      : ;
```

```
     : package DIAGNOSTIC_DRIVER is

0300 : START:  pragma FIXED_CS_ADDRESS = 16#0300#;
     :
     :     DISABLE (MICRO_EVENTS),
     :     TYP { TYP_BUS := 16#FFFF_FFFF_FFFF_FFFF# },
     :     VAL { ADDRESS_BUS := PASS_A (16#FFFF_FFFF_FFFF_FFFF#) },
     :     RESTORE_MAR_REFRESH,
     : %IF CPU_KIND.MODEL_200_CPU
     :     IOC { SET (CPU_BUSY_LIGHT) },
     : %END
     : ;
     :
0301 : SCAVANGER_INIT:
                .
                .
                .
0323 :
     :     GOTO START_VAL_TEST,
     : ;
     :
030D : START_VAL_TEST:
     :
     :     ACKNOWLEDGE_REFRESH,
     :     CALL VAL_TEST.VAL_TEST,
     : ;
030E :     GOTO START_TYP_TEST,
     : ;
     :
030F : START_TYP_TEST:
     :     ACKNOWLEDGE_REFRESH,
     :     CALL TYP_TEST.TYP_TEST,
     : ;
```

```
      : --TITLE VAL_TEST  -  Value Board diagnostic
      :
      : WITH DIAG_REGS;
      :
      : PACKAGE BODY VAL_TEST IS
      :
      :     RESULT              RENAMES DIAG_REGS.RESULT;
      :     BAD_BITS            RENAMES DIAG_REGS.BAD_BITS;
      :     PATTERN_1           RENAMES DIAG_REGS.PATTERN_1;
      :     PATTERN_2           RENAMES DIAG_REGS.PATTERN_2;
      :     EXPECTED            RENAMES DIAG_REGS.VGP_5;
      :     TEMP_1              RENAMES DIAG_REGS.VGP_5;
      :     TEMP_2              RENAMES DIAG_REGS.VGP_6;
      :     TEMP_3              RENAMES DIAG_REGS.VGP_7;
      :
      : code
      :
0400  : VAL_TEST:       pragma FIXED_CS_ADDRESS = 16#0400#;
      :
      : --
      : --  Test the FALSE condition
      : --
      : COND_FALSE:
      :     IF VAL (FALSE) THEN CALL COND_ERROR,
      : ;
0401  :     IF NOT VAL (FALSE) THEN GOTO COND_FALSE_A,
      : ;
0402  :     CALL COND_ERROR,
      : ;
0403  : COND_FALSE_A:
      :
      :
      : --
      : --  Test the TRUE condition
      : --
      : COND_TRUE:
      :     IF VAL (TRUE) THEN GOTO COND_TRUE_A,
      : ;
0404  :     CALL COND_ERROR,
      : ;
0405  : COND_TRUE_A:
      :     IF NOT VAL (TRUE) THEN CALL COND_ERROR,
      : ;
                  .
                  .
                  .
0926  :     VAL { VAL_BUS := 16#A5A5A5A5A5A5A5A5# },
      :     LOAD_WDR,
      : ;
0927  :     VAL { RESULT := WDR },
      : ;
0928  :     VAL { BAD_BITS := RESULT XOR 16#A5A5A5A5A5A5A5A5# },
      :     IF VAL (RESULT /= 16#A5A5A5A5A5A5A5A5#) THEN
      :                     RARELY CALL CMUX_WDR_ERROR,
      : ;
0929  :     VAL { VAL_BUS := 16#36C936C936C936C9# },
      :     LOAD_WDR,
      : ;
092A  :     VAL { RESULT := WDR },
```

```
     : ;
092B :      VAL { BAD_BITS := RESULT XOR 16#36C936C936C936C9# },
     :      IF VAL (RESULT /= 16#36C936C936C936C9#) THEN
     :                       RARELY CALL CMUX_WDR_ERROR,
     : ;

                        .
                        .
                        .

0ADB : COND_ERROR:
     :      DISABLE(MICRO_EVENTS),
     :      HALT,
     : ;
0ADC :      RETURN,
     : ;

                  .
                  .
                  .

0AF1 : CMUX_WDR_ERROR:
     :      DISABLE(MICRO_EVENTS),
     :      HALT,
     : ;
0AF2 :      RETURN,
     : ;
```

R1000 Command Interfaces

1. CLI

1.1. Overview

The Command Line Interpreter (CLI) interface is the lowest level interface activated on an R1000. It is available via several means:

- Initial Machine Power ON
- After a system shutdown (via BREAK or Operator.Shutdown)
- After a System Crash

1.2. Commands

This section describes the commands available from the CLI level. Generally, at the CLI command level, commands are programs which the CLI executes using the form:

```
CLI> x Command
```

where *Command* is one of the following:

```
Bootinfo        Cedit       Checkdisk
Commx           Configure   Crashdump       Crashload
Diskmd          Diskx       Display         Erasedisk
Expmon          Findseg     Gc              Initioa
Initstate       Iox         Loadee          Loader
Log             Look        Memmacs         Mt
Novram          Rdiag       Rdm             Recovery
Sam             Scan        Slew            Starter
Stat            Tapex       Trace           Update_Eeprom
```

The following are Features/limitations:

- Filenames are limited to 30 characters.
- All commands can be abbreviated.
- The following special characters are recognized:

Character	Description
*	Wildcard used in filename, match any character(s)
^P	Pop command, Forces unattached
^S	
^Q	
^H	
^C	
^R	
^U	

Figure 1.1 - CLI Special Characters

1.2.1. BOOTINFO

The BOOTINFO program is used to determine the software configuration used during the last successful system boot. It displays information about the microcode, as well as the ten subsystems which are loaded from the DFS. These subsystems are:

```
ADA_BASE
MACHINE_INTERFACE
KERNEL_DEBUGGER_IO
KERNEL_DEBUGGER
KERNEL
ENVIRONMENT_DEBUGGER
ABSTRACT_TYPES
MISCELLANEOUS
OS_UTILITIES
ELABORATOR_DATABASE
```

The information supplied for microcode is the DEC System 20 pathname of the MOM and DELTA as well as the date on which the control store image was bound. The information supplied for each subsystem is simply the name of the .MLOAD file which was used to load the subsystem into the R1000. By convention the name of the .MLOAD file provides sufficient information to find the ADA sources involved.

1.2.2. CEDIT

The CEDIT program is used to edit R1000 configurations. These configurations specify a group of microcode and software subsystems which may be loaded into the R1000 processor. In addition a configuration also specifies certain attributes of the systems hardware. Usually configurations are distributed with system software releases and should not be edited. The CEDIT program is intended for use by Rational Support Personnel when debugging or working around certain problems with new releases. Changing the contents of a configuration may make it difficult or impossible for Rational to determine which versions of software comprise the running system.

The Rational Environment is made up of several dozen subsystems. Most of these subsystems reside within the Environment's virtual memory system. Ten of the subsystems are loaded from the DFS during system boot and comprise the portions of the Environment which are required for the virtual memory system to function and for the remainder of the subsystems to be located and elaborated. These subsystems are:

```
ADA_BASE
MACHINE_INTERFACE
KERNEL_DEBUGGER_IO
KERNEL_DEBUGGER
KERNEL
ENVIRONMENT_DEBUGGER
ABSTRACT_TYPES
MISCELLANEOUS
OS_UTILITIES
ELABORATOR_DATABASE
```

For each of these subsystems a configuration file contains information about where the subsystem is within the DFS. A subsystem may be located in one of three ways:

- User is queried at load time for the subsystem. This method is only used internally for Environment development.
- The subsystem is explicitly named within this configuration. This is the normal method of locating subsystems.
- The subsystem is explicitly named in the STANDARD configuration and its location should be determined from the STANDARD. This method is useful for defining deltas from the STANDARD.

To invoke the configuration editor type:

```
CLI> x cedit
```

The program will first begin to edit the system's hardware configuration as follows:

```
Change hardware configuration [N] ?
```

If you want to modify the hardware configuration enter "Y", otherwise enter the default.

Editing Hardware Configuration

If you have chosen the default, the editor will proceed to the software configuration. If you request to modify the hardware configuration it will proceed as follows:

If the system is a Series 100 you will be asked:

```
Is this a multi-processor ? N
```

This attribute is only used on Rational's internal Series 100 processors which have been modified for multi-processing.

```
Does this processor have 8 MB memory boards ? Y
```

All Rational systems currently use eight megabyte memory modules. This question should be answered "Y".

For each of the four possible memory modules you will be asked:

```
Does memory board <n> exist ? Y
```

All Rational systems currently use all four possible memory modules. All of these four questions should be answered "Y". It is possible to run with only 3 memory modules, but not less.

Editing the hardware configuration is now done.

Editing Software Configuration

Editing the software configuration is really done in two parts. First some information about how the booting process works is needed followed by information about each of the ten subsystems. Before that the name of the configuration you are creating or modifying is needed. The CEDIT program will ask:

```
Enter name of configuration to edit [STANDARD] :
```

Enter the name of the configuration on which you want to base your changes. This is not neccessarily the configuration you will be changing. You will then be asked:

```
Enter name of configuration to save [Configuration] :
```

where *Configuration* is the default configuration name to edit.

By default the editor will save your changes in the configuration you are basing them on. You may specify a different name if you wish. If you specify a different configuration the original will not be changed.

Now the editor will inquire about some booting options. These are:

```
Allow operator to enter CLI immediately ?
```

If this question is answered "Y" the operator will be asked if he/she wishes to enter the CLI at the beginning of the booting process.

```
Allow editing of configuration ?
```

If this question is answered "Y" the operator will be asked if he/she wishes to edit the configuration being used to boot the processor during the booting process.

```
Allow operator to enter CLI prior to starting the cluster ?
```

If this question is answered "Y" the operator will be asked if he/she wishes to enter the CLI between the time that microcode is loaded and macro-code is loaded.

```
Load kernel layer subsystems only ?
```

If this question is answered "Y" only the first five subsystems will be loaded before the R1000 processor is started. This feature is used to run certain R1000 diagnostics and exercisers.

Now the editor will ask some questions about the microcode and subsystems to be used to boot the processor. These questions will be asked eleven times, once for each of the following:

```
MICROCODE
ADA_BASE
MACHINE_INTERFACE
KERNEL_DEBUGGER_IO
KERNEL_DEBUGGER
KERNEL
ENVIRONMENT_DEBUGGER
ABSTRACT_TYPES
MISCELLANEOUS
OS_UTILITIES
ELABORATOR_DATABASE
```

The questions are:

```
Take <subsystem-name> from STANDARD ?
```

This question will not be asked if you are editing the STANDARD configuration. If you answer "Y" to this question you will not be asked any more questions about this subsystem.

```
Should this configuration query for <subsystem-name> ?
```

If you answer this question "Y" then at boot time the macro-state loader will query about where the subsystem is located.

```
Enter name for <subsystem-name> ?
```

The answer to this question tells the macro-state loader where to get code for this subsystem. It is the name of an MLOAD file. If you answered "Y" to the query question the name you enter here will be the default answer when the operator is queried at boot time.

The configuration editor will now save the configurations modified during the editing process. If this succeeds it will display the message:

```
Configuration saved!
```

In any case the CEDIT program will terminate.

1.2.3. CHECKDISK

The CHECKDISK program performs a non-destructive, fast, read-only surface analysis of a formatted, labeled disk. CHECKDISK will avoid previously detected bad blocks on a labeled disk, reporting only errors which are not known.

Run the CHECKDISK program by typing:

```
CLI> x checkdisk
```

The CHECKDISK program will ask for the disk unit number to check. The disk must have been previously formatted and labeled with the RECOVERY program. Next the program will ask for the number of passes desired. Normally a single pass is sufficient but for internal or testing purposes several passes may be desired. The program will then ask if you want error information displayed for all defects or only previously-undetected defects. Normally only previously undetected error information is desired but at times all information is useful.

Once the program begins it will display the current cylinder number constantly and print a pass counter at the conclusion of each pass. The time required for a pass is dependent on the disk but may be calculated as follows:

```
((C*H*S)/16)/R seconds

where:
C = number of cylinders
H = number of heads
S = number of sectors
R = spindle revolutions per minute

Fujitsu 2351 (EAGLE) example:
((842*20*48)/16)/3961 = 12.75 minutes
```

1.2.4. CLI

The CLI is the I/O Processor's Command Line Interpreter. It is used to invoke programs, create files, delete files, display files, list files, set the time and display the time. The CLI makes use of the DFS macro user interface with which you should be familiar.

The CLI accepts the following commands:

Command	Description
COPY	Copies data from one file into another file.
CREATE	Creates a file.
DELETE	Deletes a file.
DIRECTORY	List files matching a given file specification.
LOCAL	Returns operations to the local console (issued from a remote connection).
PRINTER	Enables/disables hardcopy of console output.
REMOTE	Transfers operations to the diagnostic modem.
RENAME	Changes the name of a file.
TIME	Display/change the time.
TYPE	Displays the contents of a text file.
X	Executes a program.

Figure 1.2 - CLI Imbedded Commands

1.2.4.1. COPY

The COPY command creates a new file and copies the contents of an existing file into the new file. Command syntax is:

```
CLI> copy Existing_File New_File
```

The new file must **not** already exist. If any disk errors are encountered the command is aborted but the output file *New_File* is not deleted, and its contents are indeterminant.

Switch	Description
/D	Delete the output file if it already exists.

Figure 1.3 - COPY Switches

Example:

```
CLI> copy/d Existing_File New_File
```

where *Existing_File* and *New_File* are valid filenames.

1.2.4.2. CREATE

The CREATE command creates a file. Command syntax is:

```
CLI> create New_File
```

The new file must not already exist.

Switch	Description
/D	If a file by the given name exists, Delete the existing file before creating the *New_File*.
/SIZE=N	CREATE a file of N pages. Default = 1.
/CONTIGUOUS	Create the file with CONTIGUOUS disk space.
/I	Allow the user to enter text into the file.

Figure 1.4 - CREATE Switches

Example:

```
CLI> create/d/size=4/contiguous/i New_File
)This data will be placed into the file New_File.
))
```

The final character typed is a single ")" character on a line by itself. This terminates input and closes the file.

1.2.4.3. DELETE

The DELETE command deletes all existing files matching a given file specification. Command syntax is:

```
CLI> delete filename
```

Switch	Description
/V	Display the name of each file as it is deleted.
/C	Confirm that each file is to be deleted by asking the user.

Figure 1.5 - DELETE Switches

1.2.4.4. DIRECTORY

The DIRECTORY command is used to display information about all files matching a given file specification within the diagnostic file system. Command syntax is:

```
CLI> directory Filename
```

The size and creation time of each file matching the *Filename* is displayed along with a summary of the total number of files and total number of disk pages.

Switch	Description
/FULL	Causes information from the file's File Control Block to be displayed. This includes file attributes, allocation, and disk address information.

Figure 1.6 - DIRECTORY Switches

1.2.4.5. LOCAL

The LOCAL command is issued to a remote R1000 when customer support wishes to return control of that system to the on-site personnel.

```
CLI> local
```

See REMOTE.

1.2.4.6. PRINTER

The PRINTER command is used to enable or disable hardcopy output of all operations console transactions. Command syntax is:

```
CLI> printer on Line_Number
CLI> printer off
```

Line_number is the R1000 port number to which a line printer is attached. The printer should be on-line and should not use any software flow-control protocol. Hardware flow-control is permitted.

1.2.4.7. REMOTE

The REMOTE command is used by local personnel to transfer operations of an R1000 to Rational's Customer Support Response Center.

See LOCAL.

1.2.4.8. RENAME

The RENAME command is used to change the name of an existing file. Command syntax is:

```
CLI> rename Old_filename  New_Filename
```

Old_Filename must exist, *New_Filename* must not.

Switch	Description
/D	Delete the *New_Filename* if it already exists

Figure 1.7 - RENAME Switches

1.2.4.9. TIME

The TIME command may be used to display or set the time. It has immediate effects on the system's battery-backed-up clock/calendar. Command syntax is:

```
CLI> time                                  -- Display Time
CLI> time hh:mm:ss dd-mon-yy  -- Set the Time
```

where

HH	Hours since midnight in military (24 hour) format
MM	Minutes
SS	Seconds
DD	Day of the month
MON	One of JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
YY	Last two digits of the year

The DFS displays the time to within only 2 second granularity. The time command will set the time to the exact second.

1.2.4.10. TYPE

The TYPE command may be used to display the contents of all files matching a given file specification. Command syntax is:

```
CLI> type Filename
```

Switch	Description
/V	Display the name of each file prior to its contents

Figure 1.8 - TYPE Switches

1.2.4.11. X

Executes a program (Command) as defined in this chapter.

```
CLI> x Program_Name
```

where *Program_Name* is something like BOOTINFO, etc., as documented in this chapter.

1.2.5. COMMX

The COMMX program is a DH-11 communications multiplexer exerciser. It can be used to exercise either input or output to the R1000's RS-232C lines. To invoke the program type:

```
CLI> x commx
```

The program will query for INPUT or OUTPUT testing. Once you have answered it, proceed as follows:

```
INPUT =>
```

Any character coming into the DH-11 will be recognized and a message such as:

```
Line N on unit M received ascii XX
```

will be sent to the console as well as out the line from which the character was received. Typing any character on the console will terminate the test.

```
OUTPUT =>
```

A character string like:

```
This is line N on unit M
```

will be sent to all DH-11 lines. Typing any character on the console will terminate the test.

1.2.6. CONFIGURE

The CONFIGURE program is invoked during system boot and diagnosis to power the R1000 processor (Series 100 only), reset the R1000 processor, and verify the presence of all R1000 processor boards. It may not be invoked by the user.

1.2.7. CRASHDUMP

The CRASHDUMP program is used to capture all R1000 processor state on a magnetic tape for use in debugging system software and microcode problems. After some types of system crashes the operator will be asked if he/she wishes to take a CRASHDUMP. In most cases the operator should. This program is fairly self-explanatory but requires some input from the system operator. At times when the CRASHDUMP program is not invoked automatically the operator may invoke it as follows:

```
CLI> x crashdump
```

The CRASHDUMP program queries the operator as follows:

```
Tape unit number 0 .. 3 [0] ?
Tape density is PE(GCR), you can manually change it now.
Volume Id, (1..6 characters) ?
Please enter any comments or information that may help isolate or reproduce
this problem.  To terminate the comment, enter a ")" on a line by itself.
```

Answer these questions in the obvious fashion. The Volume ID field is used to track crash dump tapes. You should keep a log of the Volume ID you enter. The comments are of great importance when debugging from a CRASHDUMP tape. They should include processor location, your name, your phone number, and any information about what seemed to cause the system to crash, what users were on the system, and what kind of jobs were they running.

The CRASHDUMP program takes a fair amount of time to write the entire tape. The tape should be at least 1200 feet long for a GCR CRASHDUMP. The program displays the following information while running:

```
Saving state for board JKLMFQTVS [OK]
Saving Special Registers [OK]
Saving Trace Rams [OK]
Dumping CRASH_DUMP.COMMENTS [OK]
Dumping IOP_DUMP1 [OK]
Dumping IOP_DUMP2 [OK]
Dumping CRASH_DUMP.SAVED_STATE [OK]
Dumping CRASH_DUMP.REGISTERS [OK]
Dumping CRASH_DUMP.TAG_STORE [OK]
Dumping CRASH_DUMP.MEMORY 0123456789ABCDEF [OK]
Crash Dump is complete.
```

Once the CRASHDUMP is finished, remove the tape from the R1000, write-protect it, and forward it to the Rational Customer Support Response Center. Be sure to place a legible label on the tape with your name and phone number, and something to prevent the tape leader from unravelling/wrinkling during shipment.

1.2.8. CRASHLOAD

The CRASHLOAD program loads dump tapes produced by the CRASHDUMP program into an R1000 for debugging purposes. This program is intended for internal use only.

1.2.9. DISKMD

The DISKMD program is a DFS-based disk utility which is useful for a wide range of disk problems. It is USER_INTERFACE based and has several commands which act on one of two memory buffers. One buffer is the read buffer, the other is the write buffer. Each buffer is 1024 bytes in size, the same as an R1000 disk block.

Command	Description
COPY	The COPY command copies the contents of the read buffer into the write buffer.
CTS	The CTS command requires a single decimal argument (the block number) which is converted into cylinder, track, and sector and then inserted into the macro evaluation buffer.
DATA	The DATA command requires a single hex argument which is truncated to 16 bits and used to fill the write buffer.
DBN	The DBN command requires three decimal arguments (Cylinder, Track, Sector) which are converted into a disk block number and inserted into then macro evaluation buffer.
DISPLAY	The DISPLAY command displays the contents of the read buffer.
EDIT	The EDIT command requires two hex arguments. The first is used as an address in the write buffer and the second is a 16-bit data word which is inserted into the write buffer at the address specified.
RD	The RD command requires three decimal arguments which are the cylinder, track, and sector of a 1KB disk block to be read from the current unit into the read buffer. If any errors are encountered during the read, the buffer will only contain data from sectors successfully read. Other portions of the read buffer will remain unchanged.
RECAL	The RECAL command recalibrates the current disk unit.
SEEK	The SEEK command requires a single decimal argument which is interpreted as a cylinder number. The heads of the currently-selected drive are positioned to that cylinder but no transfer is done.
STATUS	The STATUS command displays the disk status from the last error for the current unit. If no errors have occurred the displayed status is indeterminate.
VERIFY	The VERIFY command compares the contents of the read buffer to the contents of the write buffer. Discrepancies are displayed on the console.
WR	The WR command requires three decimal arguments: a cylinder, track, and sector number. A 1KB transfer from the write buffer to that sector is started on the current unit. If any errors occur the transfer is aborted.
WRENABLE	The WRENABLE command is issued to write-enable the current unit. It remains in effect until the UNIT command is issued.
UNIT	The UNIT command requires a single decimal argument which is used for all disk-specific I/O from then on. The UNIT command software write protects the newly selected disk until the WRENABLE command is issued.

Figure 1.9 - DISKMD Commands

1.2.10. DISKX

The DISKX program is a DFS-based disk exerciser. It is intended to be used for quick disk subsystem integrity testing as well as long-term reliability testing. To invoke the program type:

```
CLI> x diskx
```

The program will size the system and query the operator about testing each disk unit. After these questions the exerciser will begin testing. Once testing has begun the exerciser may be stopped by typing control-G. Any other character will cause the program to display status information about each drive. This status information includes the total number of bytes transfered, total hard errors, total soft errors, and current head location.

The exerciser does transfers in the following manner:

1. Writes random data to a random block in the diagnostic region.
2. Checks the data just written.
3. Reads a random block from anywhere on the disk.

These three steps are repeated for each unit with all units running in parallel until the test is stopped.

1.2.11. ERASEDISK

The ERASEDISK program is used to comply with certain national security guidelines when removing disks from a secure site. This program will format the desired disk unit three times, using data supplied by the operator as a format pattern. The program is self explanatory. To invoke the ERASEDISK program type:

```
CLI> x erasedisk
```

The ERASEDISK program will overwrite every disk block on the disk. No information will be retained. Once the ERASEDISK program has erased a disk you will need a defects tape to re-instate the bad block region and re-format the disk. When erasing disks, it is always best to erase unit 0 last since this command is generally located on that unit.

1.2.12. EXPMON

EXPMON is the DFS-based experiment monitor. It is the basis of all low level hardware, software, and microcode debugging. The experiment monitor makes use of the macro user interface and nearly all interaction with the program is based on a collection of nearly 1000 macros. The basic command set, however, is fairly small. To use EXPMON you should be familiar with R1000 hardware, R1000 micro-architecture, R1000 macro-architecture, and the EXPMON macros.

Command	Description
XEQ	The XEQ commands causes an experiment to get loaded from the disk, parameterized, downloaded to a given board, executed, and uploaded. Once the experiment has been uploaded any output parameters are placed in the evaluation buffer. Syntax: `XEQ board-name experiment-name [parameter1,parameter2,...]`
POLL	The POLL command polls a given board and inserts the board's status into the evaluation buffer. Syntax: `POLL` *Board_Name*
RESET	The RESET command sends a reset instruction to a given board. This causes the boards DFSM to be reset. Syntax: `RESET` *Board_Name*
CONTINUE	The CONTINUE command causes an experiment running on a given board to continue from the paused state. Syntax: `CONTINUE` *Board_Name*
MEMn_EXISTS	Each of the MEMn_EXISTS (where n is 0..3) commands places a boolean into the evaluation buffer corresponding to the existence of that memory board.
QUAD_DENSITY	The QUAD_DENSITY command places a boolean into the evaluation buffer which is false if the machine has 2MB memory boards and is true if the machine has 8MB memory boards.

Figure 1.10 - EXPMON Commands

Board_Names recognized by the experiment monitor are as follows:

```
IOA     I/O adaptor  (Series 100 only)
SYS     Sysbus board (Series 100 only)
IOC     I/O channel  (Series 200 only)
TYP     Type ALU
VAL     Value ALU
SEQ     Micro-sequencer
FIU     Field isolation unit
MEM0    Memory board 0 (right-most)
MEM1    Memory board 1
MEM2    Memory board 2
MEM2    Memory board 3 (left-most)
ALL     All R1000 processor boards excluding the I/O adaptor.
```

1.2.13. FINDSEG

The FINDSEG program is used internally to determine in which subsystem a given code segment name is located. The program is invoked by typing:

```
CLI> x findseg <some code segment name>
```

If the code segment name is left off of the command line the program will prompt for one when running.

If the given segment name was loaded by the LOADER during the last system boot then FINDSEG will display the subsystem name in which it is located.

1.2.14. GC

The GC is a disk garbage collector which can delete unnecessary files within the DFS. The garbage colletor scans all existing MLOAD files and retains a list of all SEG files referenced. The program then scans all SEG files in the DFS-filesystem, deleting the ones not referenced by any MLOAD files. The garbage collector will recover disk space if unnecessary MLOAD files are deleted manually.

1.2.15. INITIOA

The INITIOA program allows the operator to change two parameters which are stored in NOVRAM on the IOA (Series 100) or the IOC (Series 200). These parameters are the *Cluster Id* and the *Phone Number* of the Rational Customer Support Response Center which is used for remote access.

Cluster Id

The system's *Cluster Id* is a unique, 6 digit, installation identifier. It is the key used to access information about the system in the Customer Support Database.

Phone Number

The PHONE_NUMBER is used by the system to begin remote diagnosis of R1000 detected problems. It is not used to make voice contact with the Rational Response Center.

The INITIOA program is intended for use during system installation by Rational Support Personel. To invoke the INITIOA program type:

```
CLI> x initioa
```

The program will repond by asking two questions:

```
Enter CLUSTER ID [<current-id>] : Cluster ID

Enter phone number to be used for remote diagnostics.
Include all PBX codes required to gain outside access.
The following characters may be imbedded at any point :
    W => Wait for dial tone.
    D => Pause 3 seconds.
    P => Subsequent digits are pulse dialed
    T => Subsequent digits are tone dialed (DTMF).
    - => No effect, used for clarity only.

Enter phone number [<current-phone-number>] : Phone Number
```

If the <current-id> is not correct, enter the correct *Cluster Id*. If the <current-id> is correct simply type <cr> to retain it.

The phone number is a character string which contains embedded characters to inform the modem about the installation site's phone system.

The following example is used to illustrate how INITIOA is used.

1. The actual phone number is 800-555-1212.
2. A PBX is in use which requires entering a "9" to gain outside access.
3. The site uses a third party phone network like "SPRINT" or "MCI" which is accessed by dialing 800-123-4567. The site's access code to this network is

121212.

If a person were making this phone call he/she would do the following:

1. Wait for the PBX dial tone
2. Dial "9".
3. Wait for the local phone systems dial tone.
4. Dial 800-123-4567 to gain access to the 3rd party network.
5. Wait for a tone.
6. Enter the access code "121212"
7. Wait for a dial tone.
8. Dial 800-555-1212 to finally contact Rational.

The modem can be instructed to do nearly the same thing. The only difference would be step 5. The modem cannot wait for a tone but instead it can be instructed to wait until it is certain that the tone has happened (by inserting 3 second delays). The modem's instructions might look like this:

```
W9W8001234567DDD121212W8005551212
```

To make this look more legible we can insert "-" characters. These are ignored by the modem. A more legible set of instructions might be:

```
W-9-W-800-123-4567-DDD-121212-W-800-555-1212
```

Once both questions have been answered the INITIOA program will store the new data and terminate.

1.2.18. LOADER

The LOADER program is used to boot the R1000. Depending on the BOOT/CRASH/MAINTENANCE options the LOADER will either prompt for a configuration to boot or simply use the STANDARD configuration.

The LOADER will first initialize the processor by invoking various experiments. Next, the microcode loader for the machine (SBUSULOAD for the Series 100 and DBUSULOAD for the Series 200) is initialized. The microcode loader will reload the control store, register file, and dispatch RAMs as needed from the microcode image specifed by the configuration file.

The LOADER will proceed to start the R1000 by invoking experiments. At this stage the R1000 will begin executing microcode to further initialize the processor and eventually become idle.

The LOADER will begin to load subsystems into the R1000 by extracting the names of MLOAD files from the configuration specified and interpreting the directions contained in them. Once all of the subsystems have been loaded into R1000 memory the LOADER will calculate the transitive closure of the subsystems, build instructions for the kernel elaborator and send those to the R1000 as well. Finally the LOADER will send a start message to the R1000. The R1000 will place a constant task name on the run queue and thus begin elaborating the environment.

Once done the LOADER invokes the monitor program which performs various tasks while the R1000 is running.

1.2.19. LOG

The LOG program allows the user to examine or initialize the DFS-based error log. This error log contains information about device errors during DFS I/O, system crash history, diagnostic failures, etc. The LOG program is interactive and is invoked by typing:

```
CLI> x log
```

Once the LOG program has been invoked it will display information about the contents of the error log. The error log is a ring buffer of 2048 entries. Newer information overwrites older information in the error log. The LOG program allows selective examination of log entries (displayed starting with the most recent first). Once you have finished examining the log you will be asked:

```
Update log header [Y] ?
```

Answering "Y" to this question will cause all entries in the error log to be marked as old. This allows subsequent examinations of the error log to filter out data already seen. Entries marked as old can still be examined until they are overwritten within the ring buffer.

To initialize the error log invoke the LOG program as follows:

```
CLI> x log initialize
```

The LOG program will ask if you really want to initialize the log. This will result in all log data being discarded. The LOG should be initialized when a new DFS is built on a system.

1.2.20. LOOK

The LOOK program can be used to examine a file within the DFS. It is invoked by typing:

```
CLI> x look Filename
```

The program will prompt for a file to examine if one is not specified on the command line. Once the file has been opened the LOOK program prompts for user input with a ">". You may now type either one or two octal arguments. The first argument is assumed to be an offset of 8-bit bytes into the file. The second word is assumed to be a count of 16-bit words to be displayed. If the second argument is not provided it is assumed to be 10 (8 decimal). The LOOK program takes the address and sets the low eight bits to zero. It then takes the count and rounds it up to the nearest multiple of eight. LOOK then displays the portion of the file specified by the modified address and count as 16-bit octal words. To terminate the program type "BYE" to the prompt. LOOK is intended for internal use only.

1.2.21. MEMMACS

The MEMMACS program provides a performance accelerator for several experiment monitor macros. It provides the following functions:

```
Logical memory read
Physical memory read
Logical memory write
Physical memory write
Tagstore display
Physical tag read
Physical tag write
Memory board state display
RDR display
TVR display
Compute ECC
```

This program should not be invoked except with the experiment monitor macros for which it is intended.

1.2.22. MT

The MT program is a DFS-based magnetic tape utility. It is used to transfer DFS files between a DFS disk and an "MT" format tape. The MT program utilizes the macro user interface. It is invoked by typing:

```
CLI> x mt
```

The following are commands executed by the MT interface:

```
Load        Dump        Rewind      Unload
```

1.2.22.1. LOAD

The LOAD command is used to transfer files from a tape to a DFS disk. The LOAD command requires no arguments and will transfer all files on the tape. The LOAD command leaves the tape positioned at End of Tape (EOT).

Switch	Description
/N	The /N switch causes the LOAD command to read the files from tape but not write them to disk. This switch may be used in conjunction with the /V switch to see what's on a tape or to position the tape at EOT to enable appending files to an existing tape.
/V	The /V switch causes the MT program to display the name of each file read from tape.
/UNLOAD	The /UNLOAD switch causes the LOAD command to unload the tape rather than leaving the tape positioned at EOT.
/UNIT=	The /UNIT= switch may be used to specify a specific tape unit for the transfer. If this switch is not present, unit zero will be used.

Figure 1.11 - MT LOAD Switches

Examples:

```
MT> load/v/unload/unit=1
```

To append files to an existing tape type:

```
MT> load/n
MT> dump file1,file2,...
```

The LOAD/N command will position the tape at EOT but not transfer any data to the disk. The dump command will then begin to transfer the specified files to the end of the tape.

1.2.16. IOX

IOX is a DFS based I/O exerciser. It is a combination of the DFS based tape exerciser, TAPEX, and the DFS based disk exerciser, DISKX. The IOX program may only be run on Series 200 processors due to memory constraints. To invoke the IOX program type:

```
CLI> x iox
```

The exerciser will prompt you as follows:

```
Simulate packet requests [Y] ?
```

Answering "Y" to this question causes IOX to use the same entry points to the I/O Processor Kernel as the R1000 would use during system operation. It provides for more extensive testing of system integrity, allows more parallelism, and increases I/O throughput.

```
Exercise Tapes [Y] ?
```

Answering "Y" to this question will cause IOX to ask if you want to exercise each tape drive which is in the system and is "ready". A tape drive is "ready" if it has a tape mounted, loaded, and write- enabled. As many as twelve tape drives may be exercised. Tape testing will destroy all data written to the tape.

```
Exercise Disks [Y] ?
```

Answering "Y" to this question will cause IOX to ask if you want to exercise each on-line disk drive. As many as sixteen disk drives may be exercised. Disk testing will only write to the diagnostic area of a disk. This should cause non-destructive results. If a disk unit is in question a full system backup should be available as certain disk malfunctions may result in writes to user data.

If you have requested that disks be exercised you will be asked:

```
Do all I/O to the same cylinder [N] ?
```

Answering "Y" to this question will cause more transfers because the disk heads will never be moved. Answering "N" will provide for testing more comparable to system usage because seeks will be quite frequent.

Once the exerciser has asked all its questions it will begin to test the devices selected. It tests the tapes and disks in the same fashion as DISKX or TAPEX. Refer to the documentation for those programs for details of testing. While IOX is running you may type **^G** (Control-G) to terminate it. Typing any other character will cause status displays which are similar to those produced by TAPEX and DISKX. Refer to those sections for details.

1.2.17. LOADEE

The LOADEE program is used to re-program the R1000 system's bootstrap Electrically-Erasable Programmable Read-Only Memories (EEPROMs). These devices contain executable data used to bootstrap the system. The EEPROMs are located on the IOA (Series 100) and on the IOC (Series 200). The LOADEE program is intended to be used by Rational Support Personnel when installing new system software releases. Documentation for the release being installed is required to successfully run the LOADEE program.

Prior to re-programming the bootstrap EEPROMs you must have loaded the desired program images into the system's DFS. To invoke the program type:

```
CLI> x loadee
```

The program will query as follows:

```
Enter I/O Processor Bootstrap file name :
Enter I/O Adaptor Cluster Manager file name :
```

Respond to these questions as instructed in the software installation instructions. The LOADEE program will transfer the contents of the specified file into the system's EEPROMs and then re-boot the R1000.

1.2.22.2. DUMP

The DUMP command is used to transfer files from a DFS disk to a tape. The DUMP command will leave the tape positioned at EOT upon transfer of the last file. The dump command requires at least one argument. This argument is the file specification to dump to tape. More than one file specification may be present on the command line. They will be processed in the order they appear.

Switch	Description
/GCR	The /GCR switch will force the file to be written in GCR format if the tape drive supports remote density selection. The /GCR switch may only be used if the tape is at BOT. GCR format records at 6250 bits per inch.
/PE	The /PE switch will force the file to be written in PE format if the tape drive supports remote density selection. The /PE switch may only be used if the tape is at BOT. PE format records at 1600 bits per inch.
/LONG_GAPS	The /LONG_GAPS switch will force the tape to be written with long or variable inter-record gaps if the tape drive supports remote gap selection. This will increase tape throughput.
/SHORT_GAPS	The /SHORT_GAPS switch will force the tape to be written with short inter-record gaps if the tape drive supports remote gap selection. This will increase tape capacity.
/THRESHOLD=	The /THRESHOLD switch helps the MT program optimize throughput. It requires a single argument. If a file being DUMPed is larger than the THRESHOLD that file will be dumped in high-speed mode. If the file is smaller it will be dumped in low speed mode. The MT program uses reasonable defaults but the /THRESHOLD allows the user to optimize further.
/UNIT=	The /UNIT= switch may be used to specify a specific tape unit for the transfer. If the /UNIT= switch is not present, unit zero will be used.
/UNLOAD	The /UNLOAD switch causes the LOAD command to unload the tape rather than leaving the tape positioned at EOT.
/V	The /V switch causes the MT program to display the name of each file read from tape.

Figure 1.12 - MT DUMP Switches

Examples:

```
MT> dump/v/gcr/threshold=25/short_gaps file1,file2,*file3*
```

To append files to an existing tape type:

```
MT> load/n
MT> dump file1,file2
```

The LOAD/N command will position the tape at EOT but not transfer any data to the disk. The dump command will then begin to transfer the specified files to the end of the tape.

1.2.22.3. REWIND

The REWIND command will reposition the tape to the Beginning of Tape (BOT).

Switch	Description
/UNIT=	The user may select a unit with the /UNIT= switch. By default unit zero is rewound.

Figure 1.13 - MT REWIND Switches

Examples:

```
MT> rewind/unit=2
```

1.2.22.4. UNLOAD

The UNLOAD command will reposition the tape to BOT and then unload the tape from the drive.

Switch	Description
/UNIT=	The user may select a unit with the /UNIT= switch. By default unit zero is unloaded.

Figure 1.14 - MT UNLOAD Switches

Examples:

```
MT> unload/unit=3
```

1.2.23. NOVRAM

The NOVRAM program is used to modify and display the NOn-Volatile RAMs located on each R1000 processor board. The NOVRAM program is used during the boot process to insure that the NOVRAMs contain valid information. The NOVRAM program may also be invoked by typing:

```
CLI> x novram
```

The program will load the contents of all boards' NOVRAMs and then display a menu. The current menu options are:

```
0 => exit
1 => modify
2 => display
```

The exit option causes the program to terminate. The modify option allows the user to modify the NOVRAM of a given board. Default answers are provided to all questions during the modification process. The default values are either the current values if the NOVRAMs checksum is correct, or best guess values if the boards NOVRAM does not checksum successfully. The display option provides a table of information about each board in the R1000 processor. Included are the board part number, serial number, artwork revision level, ECO level, and manufacturing date.

1.2.24. RDIAG

The RDIAG program is used to run FRU diagnostics. It is an interactive program based upon the macro user interface. To run RDIAG, type:

```
CLI> x rdiag
RDIAG>
```

The following commands can be executed from the RDIAG interface:

```
TEST     RUN      ERRMESS      INIT_STATE       ISOLATE
TRACE    ULOAD    MARGIN
```

and are explained in more detail on the following pages.

1.2.24.1. TEST

The TEST command is used to run all diagnostics pertaining to a given FRU.

Format:

```
DIAG> test FRU
```

Switch	Description
/1	Execute only PHASE I tests for this FRU.
/2	Execute only PHASE I and PHASE II tests for this FRU.
/3	Execute PHASE I, II, and III tests for this FRU.

Figure 1.15 - DIAG TEST Switches

1.2.24.2. RUN

The RUN command is used to execute a single diagnostic program.

Format:

```
DIAG> run [FRU] Diagnostic
```

The fru argument is only allowed when invoking diagnostics which test multiple FRUs. These include P2UCODE, P3UCODE, and all memory FRUs.

1.2.24.3. ERRMESS

The ERRMESS command is used to extract an error message from an error message file. This command is primarily intended for debugging purposes.

Format:

```
DIAG> errmess <error-message-file>,<error-number>,[<error-string>]
```

1.2.24.4. INIT_STATE

The INIT_STATE command is used to restore the R1000 processor to a known, benign, state.

1.2.24.5. ISOLATE

The ISOLATE command causes FRU diagnostics invoked with the TEST or RUN commands to invoke other diagnostics to isolate a problem.

Format:

```
DIAG> isolate {On | Off}
```

1.2.24.6. TRACE

The TRACE command enables or disables the single line of output associated with the execution of an experiment. This is most useful when debugging with a remote connection at a low baud rate.

Format:

```
DIAG> trace {On | Off}
```

1.2.24.7. ULOAD

The ULOAD command controls the loading of control-store for certain FRU diagnostics.

Format:

```
DIAG> uload {on | off}
```

1.2.24.8. MARGIN

The MARGIN command is used to margin the power and clocks utilized by the R1000 processor.

Format:

```
DIAG> margin {power | clock} {high | normal | low}
```

1.2.25. RDM Does Not Apply to 400

The RDM program is used to recover disk-defect information from new disks which have not yet been formatted. Some disk manufacturers place information about disk defects on the disk itself to allow the disk formating process to be as automatic as possible. The RDM program uses features of the SL-121 controller to retrieve this information. To run the RDM program either boot it from tape or invoke it by typing:

```
CLI> x rdm
```

The RMD program will ask several questions as follows:

```
Enter unit number of virgin disk :
```

Type the disk unit number as set on the disk drive itself.

```
Enter HDA serial number as shown on HDA :
```

Enter the disk's HDA serial number. Note that this is different from the disk drive's serial number as defects are tracked by HDA, not drive.

```
Enter the number of bytes-per-sector as jumpered at the disk :
```

This information is required to calculate the sector number in which a given defect is located. For the disks which Rational currently uses the answer to this question is:

```
Fujitsu 2351 (EAGLE)     585
Fujitsu 2361 (XP)        620
```

The answer to this question is vital. Neither the program nor the controller can determine if the information you have provided is correct. An incorrect answer will result in incorrect mapping of disk defect locations and may result is loss of user data and system downtime at a future point in time.

The RDM program will then extract information about the disk from the controller's EEPROM and display it on the console and ask:

```
Is this information correct [Y] ?
```

If these values are incorrect for the disk to be used you must run the SLEW program and change the controller's EEPROM parameters before you can recover the disk's defect information.

The RDM program will proceed to read the defect map from the disk into memory. If there are disk errors during the process the RDM program will display information about the error and retry until the defect information is recovered successfully. If the defect information is not legitimate the RDM program will inform you of the error and terminate. If the defect information is legitimate the RDM program will display each defective block in terms of cylinder, track, and sector. When the entire disk has been processed the program will display the total

number of defects found.

Once the disk has been processed the RDM program will write the defect information to either a DFS disk or MT format tape. The information will be placed in a file named:

```
<hda-serial-number>.DEFECTS
```

This file is required to format a disk with the RECOVERY program. Once the disk has been formatted the defect information recovered with RDM is no longer available. The file generated by RDM is the only source of this data and should be retained until the disk is no longer in use. The RDM program will ask about writing the defect information to either a file or tape. Answer the questions appropriately. If the RDM program was invoked from a disk it will terminate. If it was booted from tape it will restart; crash the machine to terminate it.

1.2.26. RECOVERY

The recovery program is used to prepare disks which are to be used with an R1000. The process consists of several steps which are listed here in the order required. All of these steps must be performed prior to using the disk in an R1000 system.

1. **Formatting**. The formatting process writes information onto the disk which is needed by the controller to transfer data. The format of this information can be found in Appendix A of the Spectra-Logic 121 manual. The RECOVERY program uses the controller's format drive command to format the disk. Formatting is optional when invoking the RECOVERY program unless the specified disk has no labels.

2. **Flagging bad blocks**. An integral part of building a disk for an R1000 processor is to build a bad block map which records the location of each defective area on the disk. The R1000 will not attempt to read or write to any block recorded in the bad block map. In addition to building the bad block map the RECOVERY program re-formats each bad block with the BAD SECTOR bit set in the format data. This will cause a BAD SECTOR error if any system software attempts to access these bad blocks. All bad blocks will be flagged when the RECOVERY program is invoked. This insures that blocks added to the bad block map by the system get flagged.

3. **Surface Analysis**. The surface analysis portion of the RECOVERY program is optional unless the disk was just formatted. The surface analysis is intended to insure the reliablity of the disk. From one to three passes are allowed, each taking several minutes (45 for a Fujitsu 2351 EAGLE). Each pass consists of writing to every block on the disk and then reading back each block and checking the data. A pass consists of both a write and read phase. If the disk has just been formatted the write phase of the first pass is eliminated because the process of formatting writes the data.

4. **Writing defect map**. The defect map contains information about each defective block on the disk. This data is stored on cylinder zero and is pointed to by the shared label (see below). As many as 2048 defects may be recorded with the current defect map format. The defect map is always written to the disk.

5. **Boot Label**. The boot label is located on block 1 (cylinder 0, head 0, sectors 2-3) of every disk. The boot label contains pointers to DFS files which contain the I/O Processor's Kernel and initial programs to be executed at boot time. An empty boot label is always written to the disk. It is modified as needed if a DFS is actually built on the disk.

6. **DFS Label**. The DFS label contains information about the location of the DFS directory and DFS free list. An empty DFS label is always written to the disk. It is modified as needed if a DFS is actually built on the disk. The DFS label is located on block 4 of every disk.

7. **Shared (Volume) Label**. The shared label contains information about the location of several disk structures. Some of the structures are maintained solely by the R1000 File System (RFS) and will not be mentioned here; others are shared by the DFS and the R1000 file system. These are:

   - The size of the disk. (number of cylinders, heads, sectors)
   - The location of the bad block map.
   - The location of the retarget map.
   - The location of the DFS.
   - The location of the R1000 file system.
   - The location of the read/write diagnostic portion of the disk.
   - The serial number of the disk's HDA.
   - A boolean used to indicate the presence of a DFS.
   - A boolean used to indicate the presence of an RFS.

   The shared label is always written to a disk and the portions mentioned here are only changed by the RECOVERY program. The shared label is stored using the R1000 stable storage mechanism and is located in block 2 with a copy in block 3.

8. **Building the DFS**. The DFS is only built if requested. There are several stages reported to the terminal when building a diagnostic file system. They are:

   - **Constructing free list**. The DFS retains information about free disk space as a linked list of disk extents. The free list is first constructed in memory by discarding defective blocks from the area allocated to the DFS.
   - **Writing free list**. The free list just constructed is written to the disk and its head is recorded in the DFS label.
   - **Allocating and initializing directory**. The fixed size DFS directory is allocated from the free list and initialized to be empty. At this time no files exist. Pointers to the directory are recorded in the DFS label.
   - **Allocating predefined files**. All disk structures which must have fixed disk locations or are referenced by the boot label are pre-created by the RECOVERY program. These include:
     - The disk bootstrap (located at disk block 0).
     - All bootable I/O processor kernels.
     - All bootable programs.

- All bootable file systems (series 200 only).
- The DFS error log.

9. **Loading the DFS**. After its creation the DFS may be loaded with files from an MT format tape. This step is optional.

To run the RECOVERY program you may boot it from tape or invoke it by typing:

```
CLI> x recovery
```

The program will first ask which disk drive you wish to format/build. Answer with the appropriate disk unit number. RECOVERY will then attempt to read the disk's labels and bad block map into memory. If this step fails the disk must be re-formatted and defect information read in from a tape created by the RDM program (see RDM documentation). If the labels are recovered from the disk successfully then you will be asked if the data contained in the labels should be used for the remainder of the formatting process. If you answer yes to this question then disk defect map data will be read from the disk and the disk will get built with a DFS only if it had one already.

You will be asked if you want to format the disk. Formatting the disk will destroy all data, and allow construction of a DFS on a drive which previously didn't have one.

You will be asked if you want to perform surface analysis. Surface analysis will destroy all data present on the disk. A read-only surface analysis program (CHECKDISK) can be used to check disk integrity. The RECOVERY program is not a disk test or exerciser.

Next, the disk labels will be created. If the disk had readable labels when the RECOVERY program was invoked and retained the information contained in them, that data will be used to re-build the labels. If not you will be asked to enter some information about how the disk will be used. These questions include:

```
Do you want to build a diagnostic file system on this unit [Y] ?
```

Answering yes to this question will cause space to be allocated on the disk for a DFS.

```
Enter last cylinder to be used by the DFS :
```

This question will only be asked only if the disk is to contain a DFS. The DFS will occupy all disk space between cylinder 1 and the cylinder used here. The correct answer depends on the type of disk used but should be no less than 20,000 disk blocks. It can be calculated as follows:

```
(20000 / ((H * S) / 2) + 1)

where:
H is the number of heads on the disk
S is the number of sectors on the disk
```

```
Enter first cylinder to be used for read/write diagnostics :
```

The read/write diagnostic portion of the disk starts at this cylinder and extends to the last cylinder of the disk. At least two cylinders must be allocated for read/write diagnostics. Remember that cylinder numbers start at zero, not at one; so if a disk has 842 cylinders, numbered 0 .. 841, the largest value which should be used is 840. This will cause cylinders 840 and 841 to be reserved for read/write diagnostics.

Once the disk labels have been generated the RECOVERY program is finished building the disk. If a DFS has been built on the disk you will be asked if you want to load files into the DFS. Files may be loaded from an MT format tape.

When the RECOVERY program is done, or if any unrecoverable errors occur during disk building, it will restart. The only way to terminate the RECOVERY program is by re-booting the system.

1.2.27. SAM Does Not Apply To 400

The SAM program is a programatic interface to the CDC 92185 tape drive's Structured Analysis Method of fault isolation (used with series 100 only). The SAM program may be invoked by typing:

```
CLI> x sam
```

The program is interactive and will intialize itself as follows:

```
Enter STU unit number :
```

Enter the unit number of the Streaming Tape Unit you which to test. From this point on the program will run tests which you desire and display test termination status. Some tests run forever and you will have to reset the tape drive to terminate test execution. Each SAM invocation will prompt:

```
Enter test number :
```

The program is referring to the SAM tests described in the CDC STU manual. The supported tests are 1..3, 10..26, 28..34, 37..47, 52..62, and 91, 97. Some of the tests require options. Options fall into one of four categories. They are:

```
Loop option
Bypass option
Pattern option
Speed option
```

If the test you have chosen requires options you will be prompted for them. The SAM program will then execute the selected test. Several tests do not terminate. Several other tests may not terminate if their loop option has been so set. In any case the SAM program will wait for test termination, display the termination status, and again prompt for a test to execute. To terminate the SAM program type control-C to this prompt.

1.2.28. SCAN

The SCAN program is used to search a group of text files for a given string. It is executed by typing:

```
CLI> scan
```

1.2.28.1. FIND

The FIND command requires at least one string argument. This argument is a filespec to search for a key provided via the /KEY= switch.

Switch	Description
/KEY=	Used to specify the search key.

Figure 1.16 - SCAN FIND Switches

Examples:

```
CLI> scan
SCAN> find/key=xyzzy file1,file2,*file3*
```

This command will scan FILE1, FILE2 and all files whose names contain the string FILE3. Each occurrence of XYZZY in any of these files will be displayed on the console.

1.2.29. R1000 Series 200 Models 10/20/40 PROM Debugger

The Series 200 IOC contains a 68020-based I/O Processor which replaces the PDP-11/24 used in the Series 100. To assist in various hardware and software debugging, a ROM-based, machine-level, debugger is provided. The debugger is primarily for use in manufacturing and development.

To make use of the debugger one must have fairly complete knowledge of the 68020 run-time model. The debugger's prompt is the "@" character. No type-ahead is allowed when entering debugger commands. To invoke the debugger you must place the R1000 in INTERACTIVE mode and press the BREAK key on the operations console. Then select option 3 in the menu. The debugger will be invoked immediately. Commands consist of one, two, or three characters. No carriage return is needed or allowed. Some command allow arguments which are always numbers or expresions. The radix for input and ouput may be changed. The initial radix is always 16.

Series 200 PROM Debugger Commands	
Command	Description
SD	State Display (dump all state)
RDn	Open Data register n
RAn	Open Address register n
SP	Open Stack Pointer (as defined by the PSW)
USP	Open the User Stack Pointer
ISP	Open the Interrupt Stack Pointer
MSP	Open the Monitor Stack Pointer
SR	Open the Status Register (displayed by fields)
VBR	Open the Vector Base Register
PC	Open the Program Counter
ICCR	Open the Instruction Cache Control Register
ICAR	Open the Instruction Cache Address Register
XSFC	Open the Source Function Code register
XDFC	Open the Destination Function Code register
RB	Re-Boot the IOP
RES	Software reset the IOP
expr$I	Set input radix to "expr"
$I	Display input radix
expr$O	Set output radix to "expr"
$O	Display output radix
expr$G	Set PC to "expr", and Go
$G	Go using current value of PC
expr$S	Single step through "expr" instructions
$S	Single step through 1 instruction
expr$B	Set breakpoint at address "expr"
$B	Display breakpoint list
expr$D	Delete breakpoint at address "expr"
$D	Delete all breakpoints
expr/	Open longword at address "expr"
expr\	Open word at address "expr"
expr|	Open byte at address "expr"
expr'	Open ascii character at address "expr"
^	Open previous storage unit
<LF>	Open next storage unit
<CR>	Close location
v1,V2/	Display v2 longwords starting at address v1
v1,v2\	Display v2 words starting at address v1
v1,v2|	Display v2 bytes starting at address v1

v1,v2'	Display v2 ascii characters starting at address v1
=	Display last value
expr=	Display "expr"

Figure 1.17 - Series 200 PROM Debugger Commands

Key:

```
    n           Register number 0 .. 7
    expr        An expression is a combination of numbers and
           operators.  All operators are evaluated left to
           right.  No operator precedence exists.  Allowed
           operators are:

           +    two's-complement 32-bit addition
           -    two's-complement 32-bit subtraction or
                two's-complement 32-bit negation
   32-bit one's-complement

           Where ever a number is allowed a character string
           may be used.  Character strings are enclosed in
           quotes.  In addition the "." character may be
           used in place of a number to represent the value
           of the address of the last location opened.

Examples

@123=00000123
@-123=FFFFFEDD
@~123=FFFFFEDC
@.=FFFFFEDC
@.-2=FFFFFEDA
@"ABC"=00414243
@-1+2-3+4=00000002
```

1.2.30. SLEW Does Not Apply on 400

The SLEW program is used to re-configure a Spectra-Logic SL-121 or SL-121+ tape/disk controller for use in an R1000. The controller uses an EEPROM to record several dozen options and these options are updated via SLEW. To invoke the SLEW program you may boot it from tape or type:

```
CLI> x slew
```

The program will initialize itself and then prompt:

```
Disk/Tape Controller Number :
```

Enter the controller number of the board you wish to modify. Controllers are related to devices as follows:

```
Disk    Tape        Controller
0- 3    0- 3        0
4- 7    4- 7        1
8-11    8-11        2
12-15               3
```

For the SLEW program to correctly modify a controller's EEPROM the controller must have at least one disk drive attached and on line; this is a controller limitation. The SLEW program will neither read nor write to any disks or tapes.

Once SLEW knows the controller number it begins a menu mode of operation. There are currently five menu options as follows:

```
1 => Write and verify EEPROM
2 => Verify EEPROM
3 => Display EEPROM location
4 => Modify EEPROM location
5 => Exit
```

Write and Verify EEPROM

This option allows the controller to be configured for any combination of drive types. You will be prompted to define the drive type for all 4 possible drives (0 to 3) which the controller board controls. A sample SLEW session is:

```
CLI> x slew
Disk/Tape controller number : 0

OPtions are:
    1 => Write and verify EEPROM
    2 => Verify EEPROM
    3 => Display EEPROM location
    4 => Modify EEPROM location
    5 => Exit
Enter option : 1

Enter information for unit 0
Drive type are :
    1 - Fujitsu 2351 (Eagle)
```

```
    2 - Fujitsu 2361 (Eagle XP)
    4 - Fujitsu 2333 -- 8inch, 337MB
    5 - Fujitsu 2344 -- 8inch, 690MB
     .
     .
Enter drive type : -- Select the number of the dirve from above

Enter information for unit 1
  .
  .
Is EEPROM write enabled (SW-4 open) [Y] ?
-- This switch is located on the edge of the disk/tape controller
-- board, in the middle, and can easily be accessed on Series 200
-- systems by opening the I/O cage door and reaching in.  On
-- Series 100 systems, the IOP needs to be pulled out far enough
-- to expose this portion of the board.
```

When done, make sure to set the EEPROM write enable switch back to CLOSED, and to then reboot using the *white button* reset. Failure to reset the system in this manner will result in the controller board using the older original values.

Command	Description
Write and verify	This is the option used most frequently. It updates the contents of the entire EEPROM based on questions asked of the user. The questions are asked for each of the four possible disks attached to the controller. Answer the questions carefully. Incorrect responses may prevent the system from booting after running SLEW. Once all disk drive information has been entered the SLEW program will ask: `Is EEPROM write enabled (SW-4 open) [Y] ?` The SL-121 and SL-121+ controllers have a write-protect feature. If SW-4 on the edge of the controller board is in the closed position the SLEW program cannot write to the EEPROM. Before you answer this question make sure that the switch is OPEN. If you answer 'N' to this question the SLEW program will not try to write to the EEPROM. You may use this as an escape back to the menu. Writing the EEPROM takes several seconds. It is followed by a short verify phase. Once the EEPROM has been updated, close SW-4 to write protect the data. When done with SLEW and after the different drive(s) have been installed, ALWAYS reboot the machine using the *white button* in order for the new SLEW values to take affect. Failure to do this will result in disk drive errors due to a drive of different SLEW parameters being controlled using the old SLEW parameters.
Verify EEPROM	The verify option allows the user to insure that the EEPROM checksum is correct. It does not check to insure that the data contained in the EEPROM is valid.
Display EEPROM location	The display option allows the user to examine the contents of a single EEPROM location. It is most useful for internal debugging of new EEPROM values.
Modify EEPROM location	The modify option allows the user to change the contents of a single EEPROM byte and modify the EEPROM checksum. This option is for internal use only and should not be used without a through understanding of the controller and the R1000 I/O processor kernel.
Exit	The exit option terminates the SLEW program.

Figure 1.18 - SLEW Commands

1.2.31. STARTER

The STARTER program is invoked automatically following system crashes. It is responsible for determining the course of action following a wide variety of problems. The STARTER will invoke one of the following programs depending on circumstances surrounding the crash:

```
LOADER
RDIAG
CRASHDUMP
CLI
```

The STARTER should not be invoked manually.

1.2.32. STAT

The STAT program displays directory utilization statistics for the DFS. In addition it checks for disk allocation errors and, optionally, may compact the disk. The STAT program is primarily intended for internal use. To invoke the program type:

```
CLI> x stat
```

The program will display various messages about the state of the DFS disk structures. If any error messages are displayed they should be resolved prior to using the R1000. Failure to correct DFS errors may result in user data loss and system downtime.

Interpretation of the information displayed by the STAT program requires a thorough knowledge of DFS disk structures.

1.2.33. TAPEX

The TAPEX program is a DFS-based tape exerciser. It is intended to be used for quick tape subsystem integrity testing as well as long-term reliability testing. To invoke the program type:

```
CLI> x tapex
```

The program will size the system and query the operator about testing each existing tape drive which is on-line. After these questions the exerciser will begin testing. Once testing has begun the exerciser may be stopped by typing control-G. Any other character will cause the program to display status information about each drive. This status information includes the time the test was started, the current time, total number of bytes transfered, total hard errors, total soft errors, and data errors.

The exerciser does transfers in the following manner:

1. Select a random data pattern and write a record of random length. This step is repeated a random number of times between 16 and 63.

2. Backspace a random number of records between 1 and the number of records just written.

3. Read and check the data of the records just backspaced over.

These three steps are repeated for each unit with all units running in parallel until the test is stopped.

1.2.33+ TOMBSTONE

The TOMBSTONE program is used to aid debugging R1000 microcode and hardware problems. The program will allow you to display information captured to disk when an R1000 cycles. To invoke the tombstone program:

CLI> x tombstone

```
0 => Exit
1 => Display tombstone file 1
2 => Display tombstone file 2
3 => Display tombstone file 3
4 => Display tombstone file 4

Enter option : 1

Analysis of tombstone 1 dated 09:21:48 24-JUN-99

Options are:

0 => Return to main menu
1 => Show all
2 => Show last console output
3 => Show Crash Classification
4 => Show restart output
5 => Show trace
6 => Show Cpu State
7 => Show queues
8 => Show IOP Kernel version & cluster ID
```

1.2.34. TRACE

The TRACE program is used to aid debugging R1000 microcode and hardware problems. It displays the contents of the R1000s trace rams which provide microcode execution history for the last 1024 clocks on a Series 100 or 2048 clocks on a Series 200 R1000 processor. The trace program can display data either from a CRASHDUMP which has been re-loaded into an R1000 or from a crashed machine. To invoke the trace program for CRASHDUMP data type:

```
CLI> x trace
```

To invoke the trace program for a live machine you must have previously stopped the machine using the experiment monitor. Then from EXPMON type:

```
EM> trace
```

In either case the trace program will load microcode execution history from the appropriate place into its memory buffer and display it on the console. The program is screen-oriented and provides interactive help if you type a "?".

1.2.35. UPDATE_EEPROM

This program is a DFS-based EEPROM programmer for use on the Series 200 IOC. The program provides the ability to

- Program a particular EEPROM
- Use the NOVRAM position on the IOC as an EEPROM programmer
- Do a simple write/read test of a specified EEPROM
- Verify the checksum in a specified EEPROM

To invoke the program type:

```
CLI> x update_eeprom
```

The program is menu-driven and self-prompting.

There are 4 EEPROMs on the IOC at locations K21, K19, K17, and K15. Of these, the EEPROMS in locations K21, K19, and K17 contain selftest programs, bootstrap programs, and utility programs used to boot the DFS. The EEPROM at K15 is the 'NOVRAM' that holds the board serial number, etc., placed there with NOVRAM and other data like the remote phone number placed there by INITIOA. The EEPROMs are 8192 bytes long [0..8191]. The program EEPROMs all have a common format:

```
[0000..8185]    program space
[8186]          reserved for write/read test
[8187..8189]    date code yymmdd
[8190]          coded location [17,19,21]
[8191]          checksum
```

At startup, the selftest program performs a verify operation on the checksums of the program EEPROMS. If there is an error, a message is displayed on the console and the red LED IOP ERROR is turned on. The only action here that will do anything is the WHITE BUTTON.

In principal, future program changes to the programs in these EEPROMS will be distributed on tape and this program will be used to place that new software in the EEPROMS. There will be three files of data associated with the three program EEPROMS

```
SELFTEST.HEX    ( in the EEPROM at K21 )
BOOT.HEX        ( in the EEPROM at K19 )
UTILITIES.HEX   ( in the EEPROM at K17 )
```

If the user specifies option 1 (Update EEPROM), the program will move the data from the specified file into the proper EERPOM. In the event the user wants to use the IOC as an EEPROM programmer, using option 2 will move the data from the file to the EEPROM in location K15.

2. Kernel Commands

2.1. Overview

This is the Kernel command level. Commands preceeded by a '*' are privileged commands, and can only be executed while in the privileged mode of kernel operation (see ENABLE_PRIV_CMDS).

```
 BATCH                      CHANGE_GC_THRESHOLDS      CLEAR_PROFILE
 CLEAR_PROFILES             DISABLE_JOB               ENABLE_JOB
 ENABLE_PRIV_CMDS           JOB                       JOBS
 JOB_NAME                   JOB_NAMES                 JOB_MTS
 JOBS_MTS                   LOAD                      MTSQ
 NOOP                       PROFILE                   PROFILES
 QUIT                       SET_MTS_PARAM
SET_TASK_FILTER
 SHOW_BAD_BLOCKS            SHOW_CONFIGURATION_BITS
SHOW_DISK_SUMMARY
 SHOW_ERROR_LOG             SHOW_GC_STATE
SHOW_MEMORY_UTIL
 SHOW_MTS_PARAMS            SHOW_NEXT_SNAPSHOT        SHOW_PORT_INFO
 SHOW_TASK_FILTER           SHOW_TASK_STATES
SHOW_VOLUME_SUMMARY

 TIME                      *ABORT_TASK
*BUILD_NEW_SYSTEM
*CHANGE_GHOST_LOGGING      *CREATE_CG_VPS
*CREATE_EMPTY_SPACE
*CREATE_VP                 *DEFAULTS                 *DELETE_SPACE
*DISABLE_PRIV_CMDS         *DISABLE_SUB_LOGGING
*ENABLE_SUB_LOGGING
*ENTER_DEBUG_CONTEXT       *FIND_BLOCK_REFS
*FIND_DISLOCATED_BLKS
*GO_BACK_IN_TIME           *HOGS                     *LMR
*LMW                       *PARTIAL_STARTUP
*REMEMBER_DEFECT
*ROUST                     *SET_SUB_BUFFER_SIZE      *SET_SUB_FIELDS
*SHOW_ALL_SPACES           *SHOW_CACHED_SPACES
*SHOW_CATALOG_STRUCT
*SHOW_CONFIGURATION        *SHOW_DEFAULTS
*SHOW_GC_FOOTPRINT
*SHOW_GHOST_LOG            *SHOW_HASH                *SHOW_MEMORY
*SHOW_SPACE_INFO           *SHOW_SPACE_STRUCT
*SHOW_SUB_FIELDS
*SHOW_SUB_TRACE            *SHOW_TAGS                *SHOW_UCODE_REG
*SHOW_VOLUME_STRUCT        *SHOW_VPS                 *SHUTDOWN
*START_ENVIRONMENT         *START_NETWORK_IO
```

```
*START_VIRTUAL_MEMORY
*TAKE_SNAPSHOT              *TRACE
*TRAVERSE_VM_STRUCT
*ZAP_BROKEN_SPACES          *ZERO_BLOCK
```

2.2. BATCH

This command will display what jobs are currently running in the batch system streams (queues).

Example

```
 *Kernel: batch
Stream 1       2:00
Stream 2       58:00
   225  51:03
Stream 3       50:00
   231  47:00
   234  46:56
   222  46:38
   233  45:42
```

2.3. CHANGE_GC_THRESHOLDS

Example

```
Kernel: change_gc_thresholds
VOLUME_NUMBER [1]:
THRESHOLD [START_COLLECTION]:
REMAINING CAPACITY (%) [10]:

Kernel: change_gc_thresholds
VOLUME_NUMBER [1]: 4
THRESHOLD [SUSPEND_SYSTEM]: xxx
EXPECTED VALUES ARE:
 START_COLLECTION  RAISE_PRIORITY    STOP_JOBS         SUSPEND_SYSTEM
 SPACE_04
THRESHOLD [SUSPEND_SYSTEM]:
REMAINING CAPACITY (%) [8]:
```

2.4. CLEAR_PROFILE

Example

```
Kernel: clear_profile
VPID [4]:
```

2.5. CLEAR_PROFILES

Example

```
Kernel: clear_profiles
```

2.6. DISABLE_JOB

Example

```
Kernel: disable_job
VPID [4]: 222
```

2.7. ENABLE_JOB

Example

```
Kernel: enable_job
VPID [222]: 223
```

2.8. ENABLE_PRIV_CMDS

Example

```
Kernel: enable_priv_cmds
You are enabling a set of commands which must be used
with extreme care.  They should be used only by
knowledgeable support personnel.  These commands can
easily crash/hang the machine; some can competely tràsh
the state of the machine such that you must recover the
machine from backup tapes.
Proceed [FALSE]: true
Password: secret
*Kernel:
```

2.9. JOB

Example

```
 Kernel: job
VPID [4]: 222
 Job Pri Stat CPU%  ModCt Cache   Disk  PgLim DskWts D/S JSegSz WsSiz WsLim
---- --- ---- ---- ------ ----- ------ ------ ------ --- ------ ----- -----
 222   0 I,DT    0      4     8     11   8000    130   0      5     5     0
```

2.10. JOBS

Example

```
 Kernel: jobs
Threshold [2]: xx
EXPECTED VALUES ARE:  0 ..  2147483647
Threshold [2]: 1
 Job Pri Stat CPU%  ModCt Cache   Disk  PgLim DskWts D/S JSegSz WsSiz WsLim
---- --- ---- ---- ------ ----- ------ ------ ------ --- ------ ----- -----
   4   0 R,AT    1   4104  9195  14150  65536 604891   0   1446 10997 11000
   5   0 R,AT    0     12    80     82  65536   2127   0     93    36   200
 183   0 I,AT    0     24    43     57   8000    184   0     68   146    50
 220   6 I,AT    0      2     1      9  16000    427   0      3     1     0
 222   0 I,DT    0      4     8     11   8000    130   0      5     6     0
 223   6 I,AT    0      2     1      9  16000   2240   0     16     0     0
 224   6 R,OE    0      1     3      0  16000    650   0     45    72    75
 227   6 I,AT    0      2     2      8  16000    843   0     32     0     0
 228   0 I,SV    0      3     8      9   8000    970   0     12    25    75
 229   0 I,SV    0      9    61     58   8000   7290   0   1592    72    75
 231   0 I,DT    0     47     8    172   8000    223   0      3     0    50
 232   6 I,CE    0      1     1      5  16000     25   0      4     0     0
 233   0 I,DT    0     22    44     61   8000    288   0     10     1    50
 247   6 I,CE    0      1     6      6  16000    152   0     15    36   150
 248   6 I,CE    0     42    80    300  16000   1192   0    102    74   150
 249   6 I,CE    0      1     5      6  16000     37   0      5    18   150
 250   6 I,CE    0      1     1      6  16000      4   0      0     0     0
 251   6 I,CE    0      1     0      6  16000      4   0      0     0     0
 252   6 I,CE    0      1     0      7  16000    429   0      7    10   150
 253   6 I,AT    0      2    10      1   8000   1056   0      0     1     0
 254   0 I,SV    0     21     2     66   8000     72   0     73     0     0
 255   6 I,CE    0      1     6      6  16000    132   0     65     7    10
```

2.11. JOB_NAME

Example

```
 Kernel: job_name
VPID [222]:
 Job CPU%      Root    Job Seg    Name
---- ---- -------- ---------  -------------------------
 222    0    4A8DE  10FB1503 \Mail_Check
```

2.12. JOB_NAMES

Example

```
 Kernel: job_names
Threshold [1]:
 Job CPU%      Root    Job Seg    Name
---- ---- -------- ---------  -------------------------
   4    4        0  14625502 System
   5    0        0  14626902 Daemons
```

```
 183     0     360B7  1499A902 SMP.DELTA.GURU % OP.INTERNAL_SYSTEM_DIAGNOSIS
 186     0         0  14841902 <?>
 188     0         0         0 <?>
 190     2         0  162B0101 [SMP.S_1 Editor]
 202     0         0  15F38901 [GZC.S_1 Command]
 206     0         0  1477E102 [LAP.S_1 Command]
 208     0         0  10FEE103 [GZC.S_1 Editor]
 209     0     32CD1  113B2100 Archive Server
 210     0    10E8D2  16057501 Queue Server
 211     0         0  147F8D02 [GZC.DESIGN Command]
 212     0     1A4D4  112DF900 Mail Transceiver
 213     0         0  11565900 [MLV.S_1 Command]
 222     0     4A8DE  10FB1503 \Mail_Check
 224     0         0  14928502 [SMP.S_1 Command]
 228     0     2B4E4  14652102 (Ftp Server)
 229     1     3E4E5  15E69101 Mail Dispatcher
 231     0     1C8E7  10F1F103 Design Facility (Rev3_2 release)
 232     0         0  112D5D00 *Login: 247
 233     0     428E9  15E63101 Mail Oe
 234     0     558EA  14657102 Registration job for PDL named PDL_2167
 235     0     20CEB  10F1ED03 Console Command Interpreter
 241     0         0  15E4D101 *Login: 246
 242     0         0  10F1A503 Print_Spooler
 252     0         0  112CED00 *Login: 16
 254     0     2A8FE  1462B902 "!COMMANDS.INTERNAL.DEC20.REV9_1_SPEC.UNITS.SER
```

2.13. JOB_MTS

Example

```
 Kernel: job_mts
VPID [222]:
 Job    K/S/P  Stat      CPU       CPU    Disk    Disk    WSet   WSet  Map   Run
                Age    Seconds     MS/S   DW/S   Waits    Size  Limit   To  Ratio
-------------------------------------------------------------------------------
 222   *D/I/0    54  00002.624      0.0    0.0     130      15      0  190   1.00
```

2.14. JOBS_MTS

Example

```
 Kernel: jobs_mts
 Job    K/S/P  Stat      CPU       CPU    Disk    Disk    WSet   WSet  Map   Run
                Age    Seconds     MS/S   DW/S   Waits    Size  Limit   To  Ratio
-------------------------------------------------------------------------------
   4    A/R/0  ++++  10594.728     70.8    0.0  604941   10999  11000         1.00
   5    A/R/0  ++++  05802.336      0.0    0.0    2127      36    200         1.00
 183   *A/R/0     2  00007.819     10.6    0.2     187      75    100  190    1.00
 184    T/I/6   126  00000.108      0.0    0.0       5      43      0         1.00
 190    C/R/6     2  00223.696     48.4    0.0    3250     174    150         1.00
 191    T/I/6  ++++  00013.593      0.0    0.0     573       1      0         0.99
 209   *S/I/0  ++++  00000.104      0.0    0.0      11       0      0         1.00
 210   *S/I/0  ++++  00000.094      0.0    0.0       7       0      0         1.00
 211    O/I/6  1192  00003.964      0.0    0.0      89       2     50  218    1.00
```

```
 212  *S/I/0   112  00442.230    0.0    0.0    4167     75     75          1.00
 213   O/I/6   100  00001.858    0.0    0.0      46    117     75   217    1.00
 215   C/I/6  9001  00030.487    0.0    0.0     515     94    150          1.00
 216   C/I/6     0  00022.941   61.0    0.0    1065    151    150          1.00
 217   C/I/6    83  00021.728    0.0    0.0     247    150    150          0.99
 218   C/I/6  9000  00055.883    0.0    0.0     817     19     50          1.00
 248   C/I/6  7335  00051.416    0.0    0.0    1192     74    150          1.00
 249   C/I/6  ++++  00001.438    0.0    0.0      37     18    150          1.00
 250   C/I/6  ++++  00000.046    0.0    0.0       4      0      0          1.00
 251   C/I/6  ++++  00000.026    0.0    0.0       4      0      0          1.00
 252   C/I/6  ++++  00005.782    0.0    0.0     429     10    150          1.00
 253   T/I/6  9982  00053.682    0.0    0.0    1056      1      0          0.99
 254  *S/I/0  ++++  00001.430    0.0    0.0      72      0      0          1.00
 255   C/I/6  8968  00005.236    0.0    0.0     132      7     10          1.00
```

2.15. LOAD

Example

```
 Kernel: load
Run Queue Load      => 1.16, 0.69, 0.56, 0.47
Disk Wait Load      => 0.00, 0.02, 0.04, 0.09
Withheld Task Load => 0.00, 0.00, 0.00, 0.00
Available Memory    => 18698 pages
```

2.16. MTSQ

Example

```
 Kernel: mtsq
Foreground Q
Background Q
Internal Transition Q
```

2.17. NOOP

Example

```
 Kernel: noop
```

2.18. PROFILE

Example

```
 Kernel: profile
VPID [222]: 190
 Job     Made      Made      Run       Made     Wait D   Wait C   Wait M
         Idle      Run      Total      Wait     Total    Total    Total
```

2.19. PROFILES

Example

```
Kernel: profiles
Job     Made      Made      Run       Made     Wait D   Wait C   Wait M
        Idle      Run       Total     Wait     Total    Total    Total
-------------------------------------------------------------------------
183       45        45       305        0         0        0        0
190      792       792      2374        0         0        0        0
195        6         6         7        0         0        0        0
196       11        11        20        0         0        0        0
197        2         2         4        0         0        0        0
202        5         5         5        0         0        0        0
206        6         6         7        0         0        0        0
208        2         2         4        0         0        0        0
209        0         0         0        0         0        0        0
210        0         0         0        0         0        0        0
211        5         5         5        0         0        0        0
212      171       171       408        0         0        0        0
213       35        35        89        0         0        0        0
215       24        24        64        0         0        0        0
216      642       643      2042        0         0        0        0
217      374       387      1547       13         0       14        0
218        2         2         4        0         0        0        0
```

2.20. QUIT

Example

```
Kernel: quit
EEDB:
```

2.21. SET_MTS_PARAM

Example

```
Kernel: set_mts_param
parameter name : help
parameter value [0]: 9
no such parameter
```

2.22. SET_TASK_FILTER

This command allows setting of filter attributes for use by the Show_Task_States command. By convention, the Set_Task_Filter command is used only from EEDB; this allows "quit", followed by EEDB "kernel" command to revert the task filter to

its default state.

See Show_Task_States, Show_Task_Filter.

Example

```
Kernel: set_task_filter
FILTER_KIND [BY_BLOCK_CONDITION]: xx
EXPECTED VALUES ARE:
 BY_BLOCK_CONDITION  BY_WAIT_STATE       BY_VPID
FILTER_KIND [BY_BLOCK_CONDITION]:
SELECTION_KIND [JUST_ONE]: xx
EXPECTED VALUES ARE:
 JUST_ONE   EVERY_ONE  PROMPT
SELECTION_KIND [JUST_ONE]:
BLOCK_CONDITION [SPARE_21]: xx
EXPECTED VALUES ARE:
 UNBLOCKED                    DECLARING_MODULE
 AWAITING_ACTIVATION          ACTIVATING_MODULE
 ACTIVATING_TASKS             AWAITING_TASK_ACTIVATION
 AWAITING_CHILDREN            TERMINABLE_AT_END
 BLOCKING_ON_ENTRY            DELAYING_ON_ENTRY
 ATTEMPTING_ENTRY             DELAYING
 ABORTING_MODULE              TERMINATED
 IN_FS_RENDEZVOUS             IN_WAIT_SERVICE
 DELAYING_IN_WAIT_SERVICE     BLOCKING_IN_ABORT
 DELETED                      ABORTED_WHILE_IN_MTS
 IN_MTS_RENDEZVOUS            SPARE_21
 SPARE_22                     SPARE_23
 BLOCKING_ON_ACCEPT           BLOCKING_ON_SELECT
 DELAYING_ON_SELECT           AWAITING_CHILDREN_IN_SELECT
 TERMINABLE_IN_SELECT         SPARE_29
 SPARE_30                     SPARE_31
BLOCK_CONDITION [SPARE_21]:
show tasks with this block condition [FALSE]: xx
EXPECTED VALUES ARE:
 YES    TRUE   NO     FALSE
show tasks with this block condition [FALSE]:
```

2.23. SHOW_BAD_BLOCKS

This command is used to display bad block information for a specified disk drive. Most disk drives have a certain number of defects when shipped from the factory. These are identified by the factory and provided with the drive so that those blocks are not used. During the drive's lifetime, more bad blocks will start to appear.

The Environment, when it encounters a new disk block which generates errors, will automatically add that block to the manufacturers list maintained on the disk drive, and *Retarget* that block to a known good block on disk. Thus, whenever a read or write is attempted to the origional block, the retarget block information is used and redirects the read/write to the new block.

The Manufacturers_And_System option will display all bad block information for the drive, in sorted order. The Retarget option will display just the set of

"retargeted" bad blocks, in sort order. Since a retargeting event generally causes the block to be entered in the bad block list immediately, retargeted blocks will show up in both lists.

Example

```
 Kernel: show_bad_blocks
VOLUME_NUMBER [1]:
KIND [MANUFACTURERS_AND_SYSTEM]: xx
EXPECTED VALUES ARE:
 MANUFACTURERS_AND_SYSTEM  RETARGET
KIND [MANUFACTURERS_AND_SYSTEM]: retarget
Kernel: show_bad_blocks
VOLUME_NUMBER [1]:
KIND [RETARGET]: manufacturers_and_system
blocks => 322 .. 322 || sectors => 0, 13, 20 .. 0, 13, 20
blocks => 425 .. 425 || sectors => 0, 17, 34 .. 0, 17, 34
blocks => 715 .. 715 || sectors => 1, 9, 38 .. 1, 9, 38
blocks => 1380 .. 1380 || sectors => 2, 17, 24 .. 2, 17, 24
blocks => 1492 .. 1492 || sectors => 3, 2, 8 .. 3, 2, 8
blocks => 1860 .. 1860 || sectors => 3, 17, 24 .. 3, 17, 24
blocks => 2340 .. 2340 || sectors => 4, 17, 24 .. 4, 17, 24
blocks => 2820 .. 2820 || sectors => 5, 17, 24 .. 5, 17, 24
blocks => 3300 .. 3300 || sectors => 6, 17, 24 .. 6, 17, 24
blocks => 3780 .. 3780 || sectors => 7, 17, 24 .. 7, 17, 24
```

2.24. SHOW_CONFIGURATION_BITS

Display the Boot/Crash/Maintenance Options, and some other information about power state of the processors.

Example

```
Kernel: show_configuration_bits
 IOP 0 POWER ON
 CPU 0 POWER ON
 OPERATOR MODE => INTERACTIVE
 KERNEL DEBUGGER AUTO BOOT => TRUE
 KERNEL AUTO BOOT          => TRUE
 EEDB AUTO BOOT            => TRUE
 KERNEL DEBUGGER WAIT ON CRASH    => TRUE
 KERNEL DEBUGGER DIALOUT ON CRASH => TRUE
 DIAGNOSTIC MODEM CAN DIALOUT => FALSE
 DIAGNOSTIC MODEM CAN ANSWER  => TRUE
```

2.25. SHOW_DISK_SUMMARY

There are 3 parts to the disk summary display.

1. A table
2. A list of in progress IO's

3. and some debugging information.

The table contains the following information, described by column.

- **Vol**. Stands for volume number
- **unt**. Stands for unit number. by convention unit i = volume i+1. But this correspondence is actually driven by the unit numbers selected at the drive, and can therefore be different.
- **Q Len**. Gives the number of blocks currently queued, but not yet issued to the iop.
- **IOP Len**. Gives the number of io requests which have been issued and not yet serviced by the iop.
- **Total Reads**. Gives the number of blocks read from the unit, since boot.
- **Total Writes**. Gives the number of blocks written to the unit, since boot.

The remaining columns displays error counts, since boot.

Errors

Seek Error, should be obvious. A "soft ecc" error is a data ecc error that was correctable. These cause blocks to be retargeted. A "hard ecc" error is a data ecc error that was not correctable. An "unrecoverable" error is any error which prevents the completion of the requested io; this includes hard data ecc errors; these generally hang the machine and require use of the manual disk error recovery procedure.

The list of in progress IOS might simply say "no disk io in progress". Otherwise, if display one line for each block which is currently involved in disk io (includes both queued blocks and requests waiting for response from the iop). Each line gives the "block address"; (3, 1057) means volume 2, block 1057. Note that a translation is required to get from the block number to the physical <cyl,trk,sector> address. Each line gives the "page address"; (1023, data, 259, 10234) means vpid 1023, segment kind "data", segment number 259, page number 10234. Each lines gives an "arrow" indicating the direction of the IO.

Example

```
 Kernel: show_disk_summary
DISK STATUS SUMMARY

            Q   IOP    Total      Total    Seek  Soft  Hard    Un   Total
 Vol  Unt  Len  Len    Reads      Writes   Errs   Ecc   Ecc  Recov   Errs
-------------------------------------------------------------------------
   1    0    0    0    190106      77972     0     0     0     0      0
   2    1    0    0    393305     127880     0     0     0     0      0
   3    2    0    0    243519      83758     0     1     0     0      1
```

```
   4    3    0    0     182910     124274     0     0     0      0      0

no disk IO in progress

Debugging information:
Ready_Volume mask => 0
Busy_Event_Page => ( 1023, DATA, 259, 241)
Volume_Offline_Event_Page => ( 1023, DATA, 259, 242)
```

2.26. SHOW_ERROR_LOG

Let "current log" denote that portion of the error which is stored by the kernel and not yet copied into a file in "!machine.error_logs". This command is used to display portions of the current log. Asks for line numbers. these are relative to the first line of the current log. The range of lines is displayed in chronological order when the first line number is less than the last line number. When last is greater than first, displays the log is reverse order.

The format of the entries is defined elsewhere in the documentation.

Example

```
 Kernel: show_error_log
first entry => 1; last entry => 180
FIRST [180]:
LAST [170]:
09:49:06 --- Ethernet Controller_Status EXOS CODE 0003 rxmt #1, 2 sec
09:40:10 !!! Job_Manager Bad_Job_Id Id =  253, Count = 2 Names = 2348FD 2344FD
09:40:09 !!! Job_Manager Bad_Job_Id Id =  199, Count = 2 Names = 97CC7
09:40:08 !!! Job_Manager Bad_Job_Id Id =  227, Count = 2 Names = 24E0E3
09:40:07 !!! Job_Manager Bad_Job_Id Id =  188, Count = 2 Names =
09:40:06 !!! Job_Manager Bad_Job_Id Id =  186, Count = 2 Names =
09:40:05 !!! Job_Manager Bad_Job_Id Id =  192, Count = 2 Names =
09:40:04 !!! Job_Manager Bad_Job_Id Id =  205, Count = 2 Names =
09:40:03 !!! Job_Manager Bad_Job_Id Id =  198, Count = 2 Names =
09:40:02 !!! Job_Manager Bad_Job_Id Id =  219, Count = 2 Names =
09:40:01 !!! Job_Manager Bad_Job_Id Id =  214, Count = 1 Names =
09:40:03 !!! Job_Manager Help Me Mr. Wizard!
```

2.27. SHOW_GC_STATE

Example

```
Kernel: show_gc_state
DISK daemon is not running
```

2.28. SHOW_MEMORY_UTIL

Example

```
Kernel: show_memory_util
MEMORY_SIZE => 32768

   ATTRIBUTE     CTL     TYP      Q     DATA     IMP    CODE    TOTAL
-------------------------------------------------------------------------
       DIRTY    2294    1051     48     7557     931     837    12719
    WRITEABLE   3789    1758     48    14239    1615     145    21595
       WIRED     723     396     60     1549     293     830     3852
   IN_TRANSIT      0       0      0        0       0       0        0
    PERMANENT      0       0      0     7448       0    3593    11041
 SNAPSHOTABLE      0       0      0      711       0       0      711
  RECLAIMABLE      0       0      0        0       0       0        0
       TOTAL    4013    1763     75    18844    1615    4569    30880

  ATTRIBUTE   MIN  MAX
----------------------
       DIRTY    0   13
       WIRED    0    7
 RECLAIMABLE    0    0
```

2.29. SHOW_MTS_PARAMS

Example

```
 Kernel: show_mts_params
Cpu_Scheduling     : Enabled
Disk_Scheduling    : Enabled
Memory_Scheduling  : Enabled

Percent_For_Background          : 20%
Min_ and Max_Foreground_Budget :-250 .. 250 milliseconds
Withhold_Run_Load               : 130
Withhold_Multiple_Jobs          : FALSE

Environment_Wsl            : 11000 pages
Daemon_Wsl                 : 200 pages
Min_ and Max_Ce_Wsl        : 150 .. 500 pages
Min_ and Max_Oe_Wsl        : 75 .. 750 pages
Min_ and Max_Attached_Wsl  : 50 .. 4000 pages
Min_ and Max_Detached_Wsl  : 50 .. 4000 pages
Min_ and Max_Server_Wsl    : 75 .. 1000 pages
Min_Available_Memory       : 1024 pages
Wsl_Decay_Factor           : 50 pages/5 seconds
Wsl_Growth_Factor          : 50 pages/100 milliseconds
Page_Withdrawal_Rate       : 1*640 pages/second

Min_ and Max_Disk_Load : 200 .. 250

Foreground_Time_Limit    : 1800 seconds
Background_Streams       : 3
Strict_Stream_Policy     : FALSE
Stream_Time and _Jobs 1  : 2 minutes, 3 jobs
Stream_Time and _Jobs 2  : 58 minutes, 1 job
Stream_Time and _Jobs 3  : 50 minutes, 0 jobs
```

2.30. SHOW_NEXT_SNAPSHOT

Example

```
 Kernel: show_next_snapshot
SNAPSHOT_NUMBER => 3816
```

2.31. SHOW_PORT_INFO

Example

```
 Kernel: show_port_info
PORT_MANAGER:     INPUT    OUTPUT
                  -----    ------
    BYTES....     51808   1596718
    PACKETS..     96473     88001
PORT_ID [0]: 16
OUTPUT: CLIENT => 677054; IOP IS BUSY
INPUT: CLIENT => 668862
Kernel: show_port_info
PORT_MANAGER:     INPUT    OUTPUT
                  -----    ------
    BYTES....     51857   1597240
    PACKETS..     96539     88051
PORT_ID [16]: 33
OUTPUT: NO CLIENT REGISTERED
INPUT: NO CLIENT REGISTERED
```

2.32. SHOW_TASK_FILTER

This command will show the filter settings used by the Show_Task_States command. The Set_Task_Filter command is used to set these filters.

See Set_Task_Filter, Show_Task_States.

Example

```
 Kernel: show_task_filter
FILTER_KIND [BY_BLOCK_CONDITION]: xx
EXPECTED VALUES ARE:
 BY_BLOCK_CONDITION  BY_WAIT_STATE       BY_VPID
FILTER_KIND [BY_BLOCK_CONDITION]:
  want UNBLOCKED => TRUE
  want DECLARING_MODULE => FALSE
  want AWAITING_ACTIVATION => FALSE
  want ACTIVATING_MODULE => FALSE
  want ACTIVATING_TASKS => FALSE
  want AWAITING_TASK_ACTIVATION => FALSE
  want AWAITING_CHILDREN => FALSE
  want TERMINABLE_AT_END => FALSE
  want BLOCKING_ON_ENTRY => FALSE
  want DELAYING_ON_ENTRY => FALSE
```

```
  want ATTEMPTING_ENTRY => TRUE
  want DELAYING => FALSE
  want ABORTING_MODULE => TRUE
  want TERMINATED => FALSE
  want IN_FS_RENDEZVOUS => TRUE
  want IN_WAIT_SERVICE => TRUE
  want DELAYING_IN_WAIT_SERVICE => TRUE
  want BLOCKING_IN_ABORT => TRUE
  want DELETED => TRUE
  want ABORTED_WHILE_IN_MTS => TRUE
  want IN_MTS_RENDEZVOUS => TRUE
  want SPARE_21 => FALSE
  want SPARE_22 => TRUE
  want SPARE_23 => TRUE
  want BLOCKING_ON_ACCEPT => FALSE
  want BLOCKING_ON_SELECT => FALSE
  want DELAYING_ON_SELECT => FALSE
  want AWAITING_CHILDREN_IN_SELECT => FALSE
  want TERMINABLE_IN_SELECT => FALSE
  want SPARE_29 => TRUE
  want SPARE_30 => TRUE
  want SPARE_31 => TRUE
Kernel: show_task_filter
FILTER_KIND [BY_BLOCK_CONDITION]: xx
EXPECTED VALUES ARE:
 BY_BLOCK_CONDITION  BY_WAIT_STATE       BY_VPID
FILTER_KIND [BY_BLOCK_CONDITION]: by_vpid
 0 .. 1023 => TRUE
Kernel: show_task_filter
FILTER_KIND [BY_VPID]: by_wait_state
  want PACKET_ID_WAIT => TRUE
  want PORT_WAIT => TRUE
  want TAPE_WAIT => TRUE
  want SYSTEM_BOOT_WAIT => TRUE
  want VOLUME_LOW_ON_SPACE_WAIT => FALSE
  want SNAPSHOT_WAIT => FALSE
  want PORT_INPUT_WAIT => FALSE
  want PORT_OUTPUT_WAIT => TRUE
  want TAPE_INPUT_WAIT => TRUE
  want TAPE_OUTPUT_WAIT => TRUE
  want PAGE_POOL_WAIT => TRUE
  want X25_WAIT => TRUE
  want X25_CALL_WAIT => TRUE
  want X25_INPUT_WAIT => TRUE
  want X25_OUTPUT_WAIT => TRUE
  want DEVICE_ERROR_LOG_WAIT => FALSE
  want MEMORY_ECC_WAIT => FALSE
  want PACKET_ID_LIMIT_WAIT => FALSE
  want PAGE_WIRE_WAIT => TRUE
  want KERNEL_DEBUGGING_WAIT => TRUE
  want SHORT_TERM_LOCK_WAIT => TRUE
  want TCP_IP_INPUT_WAIT => TRUE
  want TCP_IP_OUTPUT_WAIT => TRUE
  want U023 .. U031 => TRUE
  want CORE_EDITOR_WAIT => TRUE
  want COMPILATION_REQUEST_WAIT => TRUE
  want ACTION_MANAGER_WAIT => TRUE
  want ENVIRONMENT_DEBUGGING_WAIT => TRUE
  want NATIVE_DEBUGGING_WAIT => TRUE
  want WINDOW_INPUT_WAIT => FALSE
  want PIPE_INPUT_WAIT => FALSE
```

```
want PIPE_OUTPUT_WAIT => FALSE
want U040 .. NO_STATE => TRUE
```

2.33. SHOW_TASK_STATES

The Show_Task_States command is used to display some attributes of "interesting" modules. There are 2 primary module attributes which are examined to determine if a module is interesting:

- The module's Virtual Process ID (VPID)
- The module's block condition

The VPID filter indicates which VPIDs are considered interesting. and the block condition indicates which block conditions are considered interesting. If a module has an interesting VPID or block condition, it will be displayed. There is a sub-attribute, **Wait State**, which is examined when the module's block condition is one of the 2 wait state block conditions. There is also a filter for this sub-attribute. The Show_Task_Filter command can be used to diplay the current setting of the filters. And the Set_Task_Filter can be used to modify the filters.

There are some additional module attributes which always make a module look interesting, regardless of filter setting. For example, *aborted* modules are always considered interesting.

Example

```
 Kernel: show_task_states
CACHE/DISK [CACHE]:
16#820F8#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 4
16#E222004#; HELD_BY_MTS; PRI 3
16#2B4E4#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 14
16#23CD4#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 14
16#32BCD8#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 4
16#A60D9#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 4
16#374B7#; UNBLOCKED; PRI 8
16#4B0DE#; UNBLOCKED; PRI 14
16#E153C04#; DELAYING_IN_WAIT_SERVICE U031; PRI 3
16#B5DDC04#; UNBLOCKED; PRI 13
16#798CB#; ABORTED; PRI 14
16#DDF6804#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 3
16#427EC04#; IN_WAIT_SERVICE X25_WAIT; PRI 1
16#F5CD7#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 4
Kernel: show_task_states
CACHE/DISK [CACHE]: disk
Must first use the ENABLE_PRIV_CMDS command
```

2.34. SHOW_VOLUME_SUMMARY

Thresholds, garbage collection, etc is documented in the sys mgrs guide. The capacity table has a column labeled "rate blks/min". these values give the number of blocks consumed (per minute) since the last time this command was invoked. the

space_04 threshold is unused - so ignore it. the "next trigger" values give the value of unused capacity at which the next threshold will be triggered. otherwise, the display should be self explanatory.

Example

```
 Kernel: show_volume_summary
Volume Status Summary

 Vol     Total       Unused       Rate
 Num   Capacity    Capacity    Blks/Min
---------------------------------------
   1     369120      176712           0
   2     391680      101288           1
   3     391680      141456           0
   4     401280      142433           0

low space thresholds for volume 1:
    START_COLLECTION threshold at 20% (waiters exist)
    RAISE_PRIORITY threshold at 15% (waiters exist)
    STOP_JOBS threshold at 12% (waiters exist)
    SUSPEND_SYSTEM threshold at 7% (waiters exist)
    SPACE_04 threshold at 0% (no waiters)
    next trigger at 73824 blocks
low space thresholds for volume 2:
    START_COLLECTION threshold at 20% (waiters exist)
    RAISE_PRIORITY threshold at 15% (waiters exist)
    STOP_JOBS threshold at 10% (waiters exist)
    SUSPEND_SYSTEM threshold at 8% (waiters exist)
    SPACE_04 threshold at 0% (no waiters)
    next trigger at 78336 blocks
low space thresholds for volume 3:
    START_COLLECTION threshold at 20% (waiters exist)
    RAISE_PRIORITY threshold at 15% (waiters exist)
    STOP_JOBS threshold at 10% (waiters exist)
    SUSPEND_SYSTEM threshold at 8% (waiters exist)
    SPACE_04 threshold at 0% (no waiters)
    next trigger at 78336 blocks
low space thresholds for volume 4:
    START_COLLECTION threshold at 20% (waiters exist)
    RAISE_PRIORITY threshold at 15% (waiters exist)
    STOP_JOBS threshold at 10% (waiters exist)
    SUSPEND_SYSTEM threshold at 8% (waiters exist)
    SPACE_04 threshold at 0% (no waiters)
    next trigger at 80256 blocks

Debugging information:
    OUT_OF_SPACE_EVENT_PAGE_ADDR => ( 1023, DATA, 259, 504)
```

2.35. TIME

Example

3. EEDB Commands

3.1. Overview

EEDB can be run from an environment window by calling Op.Internal_System_Diagnosis, or can be used via the system console (use ^Z to get to the EEDB: prompt).

EEDB builds and maintains configurations of subsystems to be elaborated to run system programs. A number of general principals may be of interest.

A configuration is an ordered list of subsystems in the order that they are to be elaborated. Configurations are built from other configurations and share all subsystems below the branch point. For example, there is usually a MUX configuration, with OM and Dir configurations built off of it. Changing the subsystems in MUX will cause the corresponding subsystems in OM and Dir to change. There is also a copy facility that creates an equivalent configuration that is a full copy.

Most commands accept a wildcard notation. Either '+' or '*' match 0 or more characters. Wildcards are only permitted at the beginning and/or end of the argument, i.e. +.5.+ is legal, but a_+_0 won't ever match anything (or give an error message).

Subsystem names can be abbreviated using standard short names for the subsystems. For use with wildcards, the abbreviation must be followed by '.'; i.e. UAT.+ matches all Abstract_Types subsystems, but UAT+ doesn't.

Commands and some operands accept prefixes. For commands or arguments that are being specified, unique prefixes are required. For operations that display values (e.g. Abbreviations and Help), all values matching the prefix are shown as if a + were appended to the parameter.

Commands that accept lists of arguments terminate with an empty entry.

Commands and arguments can be typed on a single line, one line per token, or any combination in between. New lines will cause a prompt to be printed for the next value. Unrecognized values will produce a message; if the value is an enumeration, possible values will be printed. In either case, the prompt will be re-issued. Commands can be terminated by either ^G or ^C (must be "quoted" if running from environment).

There is a page mode that keeps output from scrolling off the terminal and allows a number of operations. See Terminal_Settings for how to set page mode, etc. Examples of the interaction with page mode:

```
-- MORE --  (n, o, q, r, s, ?) ?
N, ^G, ^C  =>  stop command
O, ^O      =>  suppress command output
S          =>  skip output between MOREs
```

3.2. Commands

A summary of EEDB commands are:

```
ABBREVIATIONS          - Print subsystem abbreviation/full-name pairs
ADD_SUBSYSTEM          - Add a subsystem to a configuration
BUILD_CONFIGURATION    - Build a configuration
COMMON                 - Highest common subsystem for two configurations
COPY_CONFIGURATION     - Copy a configuration
CHECK_CONSISTENCY      - Check consistency of database
DEFAULT_CONFIGURATION - Set the default configuration
DELETE                 - Delete a subsystem or configuration
DISPLAY                - Short form display subsystem/configuration
ELABORATE              - Elaborate a configuration
FIND_SEGMENT           - Find a segment number
HELP                   - Print a help message
INSERT_SUBSYSTEM       - No help available for INSERT_SUBSYSTEM
KERNEL                 - No help available for KERNEL
QUIT                   - Leave Command Interpreter
READ_TAPE              - Read from tape
RECLAIM_SPACE          - No help available for RECLAIM_SPACE
REMOVE_SUBSYSTEM       - Remove a Subsystem from a Configuration
REPLACE_SUBSYSTEM      - Replace a Subsystem in a Configuration
RUNNING                - Print list of currently elaborated configurations.
SET_VERBOSITY          - Set verbosity for configuration/subsystem
SHOW_DEFAULT           - Show default configuration
SNAPSHOT               - Take a snapshot
STATISTICS             - No help available for STATISTICS
TAPE_DRIVE             - Set Tape Drive Number
TERMINAL_SETTING       - Execute terminal setting command
UNELABORATE            - Unelaborate a subsystem
VDISPLAY               - Long form display subsystem/configuration
VERBOSITY              - Print verbosity settings
```

3.2.1. ABBREVIATIONS

Print subsystem abbreviation/full-name pairs. Used to show what the short names are for corresponding subsystem names. The abbreviations can be used wherever a full subsystem name can be used. Example:

```
EEDB: abbreviations ftp
FTP_INTERFACE
UFTP
```

3.2.2. ADD_SUBSYSTEM

Add a subsystem to a configuration. Adds the subsystem(s) at the top of the configuration. Subsystem must exist, not depend on anything not already in the configuration, and may not be a subsystem already present in the configuration. Example: add initialize.6.0.0d to a mux configuration that didn't have a version of initialize.

```
EEDB: add_subsystem
Existing Configuration: mux
Subsystem.Version: init.6.0.0d
Subsystem.Version:
```

3.2.3. BUILD_CONFIGURATION

Build a configuration from another configuration. All subsystems below (and including) the parent for the new subsystem are shared between the old and new configuration. Example: build a configuration off of Mux that doesn't include Initialize, but does include the native_debugger (ND).

```
EEDB: build_configuration
New Configuration: new
Existing Configuration: mux
Parent subsystem: nd
Subsystem.Version:
```

3.2.4. COMMON

Highest common subsystem for two configurations. Used to determine if two configurations are built from each other.

```
EEDB: common
Existing Configuration: mux
Existing Configuration: new
NATIVE_DEBUGGER
```

3.2.5. COPY_CONFIGURATION

Copy a configuration. Create a complete copy of a configuration. Example: notice that the new_configuration is required; it will continue asking until it gets one. If

the new configuration exists, but isn't elaborated, it will be replaced without comment.

```
EEDB: copy_configuration
Existing Configuration: mux
New Configuration:
New Configuration: xxx
```

3.2.6. CHECK_CONSISTENCY

Check consistency of database. If it prints anything, it will be messages complaining about the internal consistency of the database.

3.2.7. DEFAULT_CONFIGURATION

Set the default configuration to be booted with EEDB is elaborated. If this is set to "Base_Configuration", EEDB will boot nothing else.

3.2.8. DELETE

Delete a subsystem or configuration. Accepts wildcards. Won't delete elaborated configurations or subsystems that are part of a configuration. This last can be used to collect garbage, e.g.:

```
EEDB: delete
Subsystem/Configuration: +.+
```

This would generate a large number of messages about subsystems that couldn't be deleted, but will delete all subsystems not mentioned in any configuration (assuming that configurations don't have '.'s in them).

3.2.9. DISPLAY

Short form display subsystem/configuration. For configurations, only the name is given. For subsystems, the name and date are displayed. Configurations will all be listed before subsystems if both are applicable.

```
EEDB: di ece.+

Subsystems :
CORE_EDITOR.6.0.0D              01/09/86 17:07:00
CORE_EDITOR.5.2.3D              01/09/86 20:33:50
CORE_EDITOR.5.2.0D              12/07/85 13:28:09
CORE_EDITOR.5.2.1D              12/14/85 14:29:15
CORE_EDITOR.6.1.0D              01/15/86 13:04:52

EEDB: di a_5+

Configurations :
A_5_7_1
A_5_8_0
```

```
EEDB: display
Subsystem/Configuration: d_9_19_0

Configurations :
D_9_19_0
```

3.2.10. ELABORATE

Elaborate a configuration. Elaborates any subsystems that have not been elaborated, but are part of the configuration. After the first configuration has been elaborated, any further configurations must have been built from the same stem as the first. For test configurations, such as OM, elaborate will do nothing if the test program has completed and has NOT been unelaborated.

3.2.11. FIND_SEGMENT

Find a segment number, indicating what subsystem it comes from.

```
EEDB: find 10513433
CORE_EDITOR.6.0.0D
```

3.2.12. HELP

Print a help message for a command.

```
EEDB: help
Help for command:
ABBREVIATIONS          - Print subsystem abbreviation/full-name pairs
ADD_SUBSYSTEM          - Add a subsystem to a configuration
BUILD_CONFIGURATION    - Build a configuration
COMMON                 - Highest common subsystem for two configurations
COPY_CONFIGURATION     - Copy a configuration
CHECK_CONSISTENCY      - Check consistency of database
DEFAULT_CONFIGURATION - Set the default configuration
DELETE                 - Delete a subsystem or configuration
DISPLAY                - Short form display subsystem/configuration
ELABORATE              - Elaborate a configuration
FIND_SEGMENT           - Find a segment number
HELP                   - Print a help message
INSERT_SUBSYSTEM       - No help available for INSERT_SUBSYSTEM
KERNEL                 - No help available for KERNEL
QUIT                   - Leave Command Interpreter
READ_TAPE              - Read from tape
RECLAIM_SPACE          - No help available for RECLAIM_SPACE
REMOVE_SUBSYSTEM       - Remove a Subsystem from a Configuration
REPLACE_SUBSYSTEM      - Replace a Subsystem in a Configuration
RUNNING                - Print list of currently elaborated configurations.
SET_VERBOSITY          - Set verbosity for configuration/subsystem
SHOW_DEFAULT           - Show default configuration
SNAPSHOT               - Take a snapshot
STATISTICS             - No help available for STATISTICS
TAPE_DRIVE             - Set Tape Drive Number
TERMINAL_SETTING       - Execute terminal setting command
UNELABORATE            - Unelaborate a subsystem
```

```
VDISPLAY              - Long form display subsystem/configuration
VERBOSITY             - Print verbosity settings
```

3.2.13. INSERT_SUBSYSTEM

The same as ADD_SUBSYSTEM, except that it allows additions in the middle of a configuration. Additional parameter, parent, required.

```
EEDB: insert
Existing Configuration: mux
Parent subsystem: init
Subsystem.Version: init.5.0.1d
subsystem INITIALIZE is already in configuration MUX
```

3.2.14. KERNEL

Starts the kernel command interpreter.

```
EEDB: kernel
Kernel:
```

3.2.15. QUIT

Leave Command Interpreter.

```
Kernel: quit
EEDB:
```

3.2.16. READ_TAPE

Read from tape.

3.2.17. RECLAIM_SPACE

Actually delete code segments not associated with any subsystem version. Since multiple version can share code segments, this involves traversing all subsystems to determine which code segments can actually deleted. This should be run after a new release and after something akin to delete +.+

3.2.18. REMOVE_SUBSYSTEM

Remove a Subsystem from a Configuration.

```
EEDB: remove_subsystem
Configuration: mux
Subsystem INITIALIZE.5.0.1D to be deleted is not unelaborated
```

3.2.19. REPLACE_SUBSYSTEM

Replace a Subsystem in a Configuration. Used to install new subsystems into a configuration. Cannot be run on an elaborated configuration.

```
EEDB: replace_subsystem
Existing Configuration: mux
Subsystem.Version: init.6.0.0d
```

3.2.20. RUNNING

Print list of currently elaborated configurations. Test configurations that have not been elaborated are marked as (partial), which means that they are built off of the running configuration, but are not currently elaborated.

```
EEDB: running
  MUX
  ED (partial)
  DT (partial)
  OM (partial)
```

3.2.21. SET_VERBOSITY

Set verbosity for configuration/subsystem when displayed using VDisplay. Operation consists of setting a particular field to be displayed or not for either subsystems or configurations (as classes, not for specific instances). Possible information is:

```
EEDB: set_verbosity
Subsystem/Configuration: subsystem
Display option: ?
Possible completions for Verbosity options
ALL_CODE_SEGMENTS          MODULE_KEY
DATE                       NAME
ELAB_CODE_SEGMENT          SUBSYSTEM_DEPENDENCIES
LIBRARY                    USER
Display option: name
True or False: true
Subsystem options are now: NAME DATE USER LIBRARY SUBSYSTEM_DEPENDENCIES
    ELAB_CODE_SEGMENT
```

3.2.22. SHOW_DEFAULT

Show default configuration, i.e. the one that will be booted when EEDB is first started. Base_Configuration is the configuration containing only EEDB, which is always elaborated.

```
EEDB: show_default
Default configuration is BASE_CONFIGURATION
```

3.2.23. SNAPSHOT

Take a snapshot

3.2.24. STATISTICS

Not implemented.

3.2.25. TAPE_DRIVE

Set Tape Drive Number. If there were more than one tape drive, would allow setting the drive to be used by the tape command.

3.2.26. TERMINAL_SETTING

Execute terminal setting command. Allows setting of:

```
    COLUMNS_PER_LINE        how many columns to use for results
    ECHO_MODE               show fields as parsed; for debugging EEDB
    LINES_PER_PAGE          how many lines on the terminal
    PAGE_MODE               should output stop when more than a page
    SETTINGS                shows the values of the settings
```

```
EEDB: term
Terminal Setting: ?
Possible completions for Terminal_Command
COLUMNS_PER_LINE              PAGE_MODE
ECHO_MODE                     SETTINGS
LINES_PER_PAGE
Terminal Setting: settings
Terminal settings: lines = 24; columns = 79
EEDB: term page
Page mode: ?
Possible completions for Boolean
FALSE   TRUE
Page mode: false
```

3.2.27. UNELABORATE

Unelaborate a subsystem.

3.2.28. VDISPLAY

Long form display subsystem/configuration. The line of dashes give information about the configuration that makes up EEDB. Subsystems with the sequence number .XXX are not registered with EEDB.

```
EEDB: vd mux

Configurations :
MUX
```

```
    INITIALIZE.5.0.1D              12/11/85 12:19:28 Key: 1AFF3C04
    NATIVE_DEBUGGER.5.1.5D         12/15/85 18:14:44 Key: 1AFD8004
    ARCHIVE.5.0.5D                 12/11/85 23:43:04 Key: 1AFD2404
    ...
    OM_MECHANISMS.5.0.2D           12/26/85 14:51:16 Key: 1ACA2804
    TEST_UTILITIES.4.0.1D          09/16/85 19:06:56 Key: 1AC9E004
    NETWORK.5.0.3D                 11/18/85 08:37:36 Key: 1AC96004
-------------------------------------------------------------------------
    ELABORATOR_DATABASE.5.0.0D     11/08/85 14:27:20 Key: 1AC80C04
    OS_UTILITIES.5.1.0D            12/06/85 13:03:07 Key: 1A83CC04
    ...
    MACHINE_INTERFACE.4.0.1        09/03/85 15:41:02 Key: 04002C04
    ADA_BASE.4.1.0                 10/14/85 12:53:40 Key: 00010004
    MICROCODE.4.92                 12/16/85 16:54:01

EEDB: vd init.5.0.1d

Subsystems :
INITIALIZE.5.0.1D              12/11/85 12:19:28 DRK
    Lib:  :NET:CURLY:PDD:INITIALIZE.5.0.1:LIBRARIES:INITIALIZE.LIB
    With: R1000_CODE_GEN                DIRECTORY
          BASIC_MANAGERS                KERNEL
          KERNEL_DEBUGGER               OS_COMMANDS
          COMMANDS                      MACHINE_INTERFACE
          INPUT_OUTPUT                  MISCELLANEOUS
          OM_MECHANISMS                 CORE_EDITOR
          ABSTRACT_TYPES                ADA_MANAGEMENT
          PARSER                        ELABORATOR_DATABASE
          ADA_BASE                      TOOLS
          ENVIRONMENT_DEBUGGER
    Elab: 1809431
    Code: 252951    231447    10763279  251927    1809431   1808407
```

3.2.29. VERBOSITY

Print verbosity settings.

```
EEDB: verbosity
Subsystem fields to be displayed when printing
Configurations :  NAME DATE USER MODULE_KEY
Subsystems :      NAME DATE USER LIBRARY SUBSYSTEM_DEPENDENCIES
    ELAB_CODE_SEGMENT
EEDB: set_verbosity
Subsystem/Configuration: configuration
Display option: ?
Possible completions for Verbosity options
ALL_CODE_SEGMENTS          MODULE_KEY
DATE                       NAME
ELAB_CODE_SEGMENT          SUBSYSTEM_DEPENDENCIES
LIBRARY                    USER
Display option: user
True or False: true
Configuration options are now: NAME DATE USER MODULE_KEY
```

4. Procedures for System Hang Condition

To perform preliminary diagnosis and provide the maximum amount of information for diagnosis of a system crashdump tape, the following procedures should be performed prior to crashing the system and taking a crashdump. If possible, Rational should be contacted prior to crashing a "hung" system. Whenever a crashdump tape is taken, it is very important to include as much information about the crashdump as possible. The following procedures describe the minimal amount of information necessary based on the type of the system hang.

No System Console Response

The system console might be flow controlled, which will make it appear as if the system is hung.

1. Does the console respond to the BREAK key with the "Please enter 0/1/2" message? If not, try typing "2", followed by a <CR>. If the console now responds to the BREAK key, then the console was flow controlled by the IOA/C waiting for the "0/1/2" response. If the console still does not respond, then there is a problem with the console terminal, the connection to the IOA/C, the IOA/C itself, or another piece of hardware. Verify that the terminal is functioning correctly (power the terminal OFF, then ON to initiate self testing). In the event that no response can be obtained from the R1000, a crashdump tape will not be useful, and Rational should be called immediately to coordinate further diagnosis of the problem.

2. Does the console switch banners in response to ^Z? If not, try typing <CR>. If there is still no response, then type ^Q. If the console now responds to ^Z or <CR>, then the console was flow controlled by R1000 or IOP/C software. If the console still does not respond, then crash the system (using the BREAK key) and generate a crashdump tape indicating in the comments that the console would only respond to the BREAK key, and that this step had been reached.[1]

System Console Responds

Having reached this point, it has been determined that the system console will respond. If the prompt on the console is "CLI>" or "Enter configuration to boot..", then the machine has crashed, and the cause of the crash was displayed in previous output to the console. If this output is still visible, examine it to determine the cause of the crash and what was recommended as appropriate action. If a crashdump tape was prompted to be taken, then take one now and enter the displayed reason for the crash.

It is important to note that:

[1] A problem probably exists in Microcode or Kernel software.

1. Both interpreters (Kernel and EEDB) can present the "Kernel:" prompt.

2. By typing <CR> and using ^Z one can cycle through the prompts to determine which interpreter is displaying the "kernel:" prompt (indicated in the banner information).

3. To get the "Kernel:" prompt via the EEDB interface, type "kernel" at the EEDB prompt.

4. It is always better to use the kernel prompt under the EEDB banner. If the EEDB becomes hung, the non-faulting Kernel interpreter is still available.

Can the Kernel or EEDB command interpreters be reached? If not, gather the following information:

1. The output produced by BREAK-2.

2. The banners displayed by ^Z.

3. The prompts that are displayed on the console (use ^Z).

then crash the system (using the BREAK key) and make a crashdump tape, including this information along with the crashdump tape.

Having reached this point, it has been determined that the R1000 is still running, and that the Kernel/EEDB interpreters respond.

1. Do "Show_Disk_Summary" at the Kernel prompt. Does it show unrecoverable disk errors? If so, call Rational. This is most likely the reason for the hang.

2. Does "Show_Disk_Summary" show IO in progress? If so, execute the command several times. If the successive displays show no change in the total read/writes columns, yet the display shows disk IO in progress, there is an IO hang. Execute a "Show_Task_States", record the information and submit it along with a crash dump tape.

3. Do "Show_Volume_Summary". If any volume shows 0 unused capacity, then the system has reached the suspend system GC threshold. This can be verified by noting that no waiters exist for the suspend threshold for that volume. Do not submit a crash dump. Reboot following the procedures in the System Manager's Guide for recovering when the Suspend_System threshold has been reached.

4. Do a "Jobs" (threshold 1) command. Do executions of this command show activity in the CPU and D/S categories? There should be either virtually no activity, or some job(s) consuming virtually all cycles.

5. If the problem is too much activity, wait 5 or 10 minutes before proceeding. If the activity is in user jobs, then disable the user job, and do not submit a crashdump. Determine what the user job is doing and take the appropriate action. If the activity is in job 4 or 5, record several executions of the "Show_Task_States" (Cache) and submit this information along with a

crashdump tape.

Having reached this point, the R1000 is still running, the Kernel/EEDB interpreter respond, and the system appears to be idle, yet it also appears to be hung.

1. Use "Show_Error_Log" and search for anything out of the ordinary.

2. Can users login? If not, try more than one connection. If telnet is being used, try a serial (DH11) port (like 16).

3. Are users logged in but getting no response? Are all users having this problem? It is possible for a user's session to become hung, which doesn't imply that all sessions are hung. If not all user's sessions are hung, examine the session error logs (in !Machine.Error_Logs) for the hung sessions. Also do a "Show_Task_States" (Cache) and see if any task is in an "Environment_Debugging_Wait". If so, execute the Show_Tasks procedure in the System_Maintenance subsystem, using the task ID of the task shown to be in the "Environment_Debugging_Wait", and call Rational with the output of this command.

Having reached this point, the most effective means of debugging the problem is to contact Rational and have a remote debugger connected to the system. A crashdump tape will generally prove inconclusive.

Failure to provide the information described above along with a crashdump tape will generally result in an undiagnosible crashdump tape. A description of simply "system hang" is not sufficient to diagnosis a system hang.

Table Of Contents

Table Of Figures

R1000 – Diagnostics and Maintenance

1. Physical Maintenance

There is little physical maintenance required for the R1000. Those items that should be checked on a periodic basis are;

- Fan Filters
- Tape Drive
- Cable Connections
- Clearance

Fan Filters – the filters are under the CPU cage and should be inspected and cleaned on a regular interval. The interval required is a function of the air quality for the location of the system. In a clean environment the interval between checking will be longer. In general checking the fans twice a year should be sufficient. This work should be scheduled with a PM or some other time when the system will be shutdown.

To check and clean the fans, open the front door to the machine. Under the CPU bay is a small tray with contains the fans and two filters. You can only get this tray out after you disconnected the power cable for the fans. Each filter is connected to this fan tray with four little screws. After the filters have been removed from the tray, you can put the tray back in so the fans can do their work.

- The machine might crash if you disconnect/reconnect the power cable for the fans
- The machine might crash due to overheating if you leave the fans out of the machine too long.
- If you plan to take the risk of a crash, make a snapshot before you connect/disconnect anything

Again because of the possibility of a crash, it is best to clean the filters during the PM when the machine is down anyway.

If you are not sure whether the filters need to be cleaned, just clean them and compare a cleaned filter with one you didn't clean yet. Hold them against the light and you will see the difference. Even filters that seem to be quite clean can in fact be rather clogged up! New filters have known to get clogged up within a year given the right circumstances!

Three methods of cleaning have been applied thus far:

- Use a vacuum cleaner to get the dust out of the filters.
- Use high pressure air and blow through the filters from the other side. This will create a room full of dust.

- Use hot water with a bit of soap. You need to let the filters dry before you put them back in the machine.

Make sure you have the means to clean the filters before you take the machine down.

Tape Drive – The vendor recommendations are -

> The EXB-8200 tape head/path requires cleaning once a month or after 30 gigabytes of data transfer, whichever occurs first. For planning purposes, 1 gigabyte of data is transferred for every hour of continuous streaming operation. -- EXB-8200 Product Specification

One does need to keep in mind that the cleaning process is abrasive. Over cleaning of a drive can shorten its lifetime.

Cable Connections – During a PM or other period when the system will be shutdown the cables should be check to confirm that they are still properly seated.

Clearance – The system should be inspected to ensure that no items are blocking air flow. Efforts should be made to not set items on top of the system, as this can block vents, reducing airflow that could lead to overheating.

2. Preventative Maintenance (PM)

The PM is an extension of Physical Maintenance adding the review of systems resources and running of diagnostics.

PM steps are normally performed by Rational personnel. However for system not under support the customer can perform all or some portion of the maintenance.

The initial step is to generate the PM report.

!Tools.System_Availability.<rev>_Spec.Units.Reports.Pm_Report

```
-- This procedure will generate a Preventive Maintenance (PM) Report.  This
-- report provides useful system information for analysis and history for
-- later reference.  The period defined by Starting_Date to Ending_Date is
-- used when summarizing the log file entries.  If Delete_Log_Files is
-- TRUE, then those log files that fall between Starting_Date and
-- Ending_Date are deleted from !Machine.Error_Logs.  If Print_Report is
-- TRUE, then the report is printed via the system default print queue.
-- The default Report_Filename is a file created in the current context,
-- called PM_Report_<Date/Time>.  This procedure is designed to execute in
-- as a background job, and will cause this to occur immediately after it
-- begins to execute.
procedure Pm_Report (Starting_Date : String := "<MINUS_90_DAYS>";
                     Ending_Date : String := "<TODAY>";
                     Delete_Log_Files : Boolean := False;
                     Print_Report : Boolean := True;
```

```
Report_Filename : String := "<DEFAULT>";
Log_Directory : String := "!Machine.Error_Logs";
Accounting_Library : String := "!Machine.Accounting");
```

The report generated gives both information on systems resources as well as a checklist of steps and verification to perform on the system.

Analysis of the system can also be performed using

!Tools.System_Availability.<rev>_Spec.Units.System_Report.Generate

A programmatic interface is available through

!Tools.System_Availability.<rev>_Spec.Units.System_Information

Date : May 23, 1999 at 4:11:35 PM
Customer : ***
Machine : r1000_one [444990]
Period : 22-FEB-99 to 23-MAY-99

RATIONAL
PREVENTIVE MAINTENANCE CHECKLIST

(1) Overall System Health Check ___

Ensure customer has a current FULL backup ___
Review system error logs summary ___
Run 5 minutes disk exerciser ___
Run 1 pass tape exerciser ___
Run Level 2 FRU diagnostics ___
Check +5v DC at R1000 backplane _____v _____v _____v ___
Check 15v DC _____v _____v (Series 200) ___

(2) General System Cleanliness ___

Check/clean main cabinet filters ___
Check/clean disk filters ___
Clean cabinets and tape drive surfaces ___
Clean inside and top of R1000 cabinets where necessary ___
Check all fans for proper operation ___
PM line printer ___
Page Counter _____pages ___
Check/replace ribbon ___
Clean ribbon rollers ___
Clean print band ___
Check for character alignment and print quality ___
Vacuum ribbon lint and paper dust ___
PM tape drive
Clean R/W head, tape cleaner, and tape path ___
Clean filters on IOP (Series 100) ___
Clean operations terminals ___

(3) Post PM System Diagnostics ___

Run foreplane test ___
Run full pass of R1000 processor diagnostics ___
Run functional diagnostics ___
Run 01 test on tape drive with full (2400 ft.) tape ___
Run 5 minutes disk exerciser ___
Run 1 minute tape exerciser ___
Run communications test on at least 1 port per 8-port module ___

(4) Complete SAR ---

```
Booted Configuration : D_12_7_4
Machine Id = 444990

R1000 Series 400 System

Board                  Part#   Serial#    Art Rev   ECO Level  Build Date
---------------------- ------- ---------- --------- ---------- -------------
Sequencer              5       0          2         5          1/03/05
Field Isolation Unit   44      12         1         1          90/08/28
IOC                    9       4386       2         10         92/03/30
Type                   43      4652       2         4          91/05/29
Value                  42      4094       1         3          92/11/25
32 MB Memory           47      4212       2         0          91/03/26
32 MB Memory           47      3995       2         0          90/11/11
ReshA                  41      2          1         5          90/08/10
CMC controller serial number = 526
System Boot Configuration Settings
==================================
Maintainance_Mode => True
Kernel_Debugger_Auto_Boot => True
Kernel_Debugger_Wait_On_Crash => True
Kernel_Debugger_Dialout_On_Crash => True
Modem_Can_Dialout => False
Modem_Can_Answer => False
Eedb_Auto_Boot => True
Kernel_Auto_Boot => True


         DISK BAD BLOCK INFORMATION
Vol  Manufact   Retarget   Total     HDA
=========================================
 1       127         0       127   J2452
 2       149         0       149   J2586
 3       112         0       112   J2587
 4        90         0        90   23423424

Daemon Information Display
--------------------------

Shows at end of day final run of daemon.  This
indicates size before compaction and final size
in pages.


  Date        Ada            File          Action     Directory        DDB
========  ============  ==============  ========  ==============  ==========
99/04/18                                44 -> 44                  691 -> 688
99/04/19  6447 -> 6400  10294 -> 10270  44 -> 44  15255 -> 15196  691 -> 688
99/04/20  6401 -> 6400  10271 -> 10270  44 -> 44  15212 -> 15196  691 -> 688
99/04/21  6401 -> 6400  10271 -> 10270  44 -> 44  15212 -> 15196  691 -> 688
99/04/22  6401 -> 6400  10271 -> 10271  44 -> 44  15212 -> 15197  691 -> 688
99/04/23  6401 -> 6400  10271 -> 10271  44 -> 44  15212 -> 15197  688 -> 688
99/04/24  6401 -> 6400  10272 -> 10271  45 -> 45  15213 -> 15197  688 -> 688
99/04/25  6401 -> 6400  10274 -> 10271  41 -> 41  15213 -> 15197  691 -> 688
99/04/26  6445 -> 6400  10296 -> 10272  41 -> 41  15273 -> 15198  691 -> 688
99/04/27  6400 -> 6400  10273 -> 10272  41 -> 41  15214 -> 15198  691 -> 688
99/04/28  6400 -> 6400  10273 -> 10272  41 -> 41  15214 -> 15199  691 -> 688
99/04/29  6400 -> 6400  10273 -> 10272  41 -> 41  15214 -> 15199  691 -> 688
99/04/30  6401 -> 6400  10273 -> 10272  42 -> 42  15215 -> 15199  691 -> 688
99/12/25  6401 -> 6400  10273 -> 10272            15215 -> 15199  691 -> 688

.
.
.
```

3. Diagnostics

For the R1000, there are two forms of hardware diagnostics. The first are the board based diagnostics or otherwise know as "experiments" that together have come to be known as the FRUs. The second form of hardware diagnostic is the Micro-Diagnostics or udiag. The second is a Control Store image that implements a suite of tests to qualify the hardware.

The FRUs are single clocked, and run tests local to a specific board. The FRUs can be selected as option 3 on the startup menu, or can be launched from the CLI prompt.

```
CLI> x fru
Rational R1000 FRU diagnostic driver
Initializing ...LOAD_SYSBUS_STATE.SYS  RUN_SYSBUS_ONLY.SYS  RESET.SYS
READ_NOVRAM_DATA.IOA  READ_NOVRAM_DATA.
SYS  READ_NOVRAM_DATA.TYP  READ_NOVRAM_DATA.VAL  READ_NOVRAM_DATA.FIU
READ_NOVRAM_DATA.SEQ  READ_NOVRAM_DATA.MEM
  READ_NOVRAM_DATA.MEM  READ_NOVRAM_DATA.MEM  READ_NOVRAM_DATA.MEM  done!

Main menu
    1 => Display cluster information
    2 => Execute confidence tests
    3 => Execute diagnostics
    4 => Execute PM tests
    5 => Margin cluster clocks/power
    6 => Specify test parameters
    7 => Repair assistance
    8 => Initialize processor state
    0 => Exit
Please enter option : 3

Diagnostic execution menu
    1 => Test the foreplane
    2 => Run all tests
    3 => Run all tests applicable to a given FRU
    4 => Run a specific test
    0 => Return to main menu
Please enter option : 3

Fru menu
    1 => ALL                          7 => FIU
    2 => Val ALU                      8 => Memory 0
    3 => Typ ALU                      9 => Memory 1
    4 => Sequencer                   10 => Memory 2
    5 => Foreplane                   11 => Memory 3
    6 => SYSBUS                      12 => I/O Adaptor
Please enter option : 1
Please enter maximum test phase (1-3) : 2
```

Would produce output of the form.....

```
Running FRU P1DCOMM
Running FRU P1IOC
Running FRU P1VAL
Running FRU P1TYP
Running FRU P1SEQ
Running FRU P1FIU
Running FRU P1MEM
Running FRU P1SF
Running FRU P2IOC
Running FRU P2VAL
File  : ...
Bound : ...
Delta : ...
Mom   : ...
```

The micro-diagnostic differs from the experiments based FRUs;

- "Climbs" onto the Hardware, first testing basic operation, and then moving on to test more and more complex functions
- Runs "at speed", which is to say it executes at the normal clock rate, rather than the single step approach of the FRUs.
- The Microdiagnostic is broken into sections that focus on testing a specific board. On a failure the halting address will give a good indication of the board that is at fault. By using the EXPMON to check register results and trace information it is possible to get a further understanding of the failure, possibly down to the failing component.

While considered separate from the FRUs the micro-diagnostic is run from the FRU menu, under the heading of "confidence tests" – option 2.

```
Main menu
      1 => Display cluster information
      2 => Execute confidence tests
      3 => Execute diagnostics
      4 => Execute PM tests
      5 => Margin cluster clocks/power
      6 => Specify test parameters
      7 => Repair assistance
      8 => Initialize processor state
      0 => Exit
  Please enter option : 2
```

Which will run the udiag and then report whether it succeeded or encountered a problem.

It is also possible to run the diagnostic under manual control through the experiment monitor (EXPMON)

```
CLI> boot

Enter name of configuration to boot [STANDARD] : diag
Enter CLI [N] ?
 Microcode file [diag] :
 Loading Register Files and Dispatch Rams .... [OK]
 Loading Control Store ............ [OK]
Do you want to enter the CLI prior to starting the machine [N] ? y

CLI> x expmon

     2  32MB MEMORY BOARDS IN PROCESSOR - TOTAL OF 64 MEGABYTES

EM> rd

      The EM prompt will return immediately but WAIT for about one minute.
      The diagnostics must run on each board in parallel.

EM> poll_all
      NO MACHINE CHECKS DETECTED
```

```
      The message, "NO MACHINE CHECKS DETECTED", means you did not wait long
      enough.  Wait a little longer and try again.  Stop running Poll_All when
      you find that a board has detected a machine check.  For example:

EM> poll_all
      SEQ HAS DETECTED A MACHINE CHECK

EM> sm
    HALT - NEXT UPC IS 0103
  or
    UCODE HALT AT 102

EM> bye
CLI>

   If the next UPC is 0103 (or current 102) it means that no problems were found.
   A UPC of 0101 usually means that you forgot to boot the diag configuration.  Any
   other UPC indicates a problem and may be displayed with a diagnostic
   message indicating the possible problem.

   Write down the next UPC value and any diagnostic messages.
```

When the micro-diag does not halt at 102/103, then the halting address can be mapped to the area of the diag under execution at the time, and with the a estimate of the failing component.

While still in the EXPMON you can get a trace of the most resent cycles of the system with the "trace" command.

```
EM> trace

15     2EC3
14     2EC3
13     010A
12     010B
11     2EA8
10     2EA9
 9   C 2EAF
 8     2EB0
 7   C 2EB6
 6     2EA1
 5     2EA2
 4     0321
 3     0101
 2     0103
 1   C 0102
 0     0103
```

Use "x" to exit the trace in expmon.

Basic EXPMON commands for system analysis

cfh – continue from halt

cins – Current Macro Instruction

csas – Display Control Stack Accelerated registers

dq_head – first task on delay queue

dw_head – first task on disk wait queue

faq_head – first task on Fault Awaiter queue

gp – General Purpose registers (0-15)

ibuff – Macro Instruction Buffer

mars – display memory address registers

pc – current micro-pc

pfmar – Page Fault MAR (Memory Address Register)

poll_all – poll boards to see if any halts have occurred

prep <micro_address> - Set micro PC to specified address

qsucc – traverse a linked list of tasks

rd – prep to address to start microdiagnostic and start it. This requires that the micodiagnostic be loaded into Control store

rdrs – display read date registers

rf – display elements of the Register File (0-#3FF)

rm – run machine

sm – stop machine

tlc – loop counter on TYP board

trace – display micro-pc trace

tt – display Control Top register

ustacks – micro code call stack

vlc – loop counter on VAL board

wdr – display write data registers

```
      : package DIAG_REGS is
      :
      :     PASS_COUNTER  : VGP(0);
      :     RESULT        : GP(1);
      :     BAD_BITS      : GP(2);
      :     PATTERN_1     : GP(3);
      :     PATTERN_2     : GP(4);
      :     VGP_5         : VGP(5);
      :     TGP_5         : TGP(5);
      :     GP_5          : GP(5);
      :     VGP_6         : VGP(6);
      :     TGP_6         : TGP(6);
      :     GP_6          : GP(6);
      :     VGP_7         : VGP(7);
      :     TGP_7         : TGP(7);
      :     GP_7          : GP(7);
      :     VGP_8         : VGP(8);
      :     TGP_8         : TGP(8);
      :     GP_8          : GP(8);
      :     VGP_9         : VGP(9);
      :     TGP_9         : TGP(9);
      :     GP_9          : GP(9);
      :     VGP_10        : VGP(10);
      :     TGP_10        : TGP(10);
      :     GP_10         : GP(10);
      :     VGP_11        : VGP(11);
      :     TGP_11        : TGP(11);
      :     GP_11         : GP(11);
      :     VGP_12        : VGP(12);
      :     TGP_12        : TGP(12);
      :     GP_12         : GP(12);
      :     VGP_13        : VGP(13);
      :     TGP_13        : TGP(13);
      :     GP_13         : GP(13);
      :     VGP_14        : VGP(14);
      :     TGP_14        : TGP(14);
      :     GP_14         : GP(14);
      :     VGP_15        : VGP(15);
      :     TGP_15        : TGP(15);
      :     GP_15         : GP(15);
      : end DIAG_REGS;
      :
0100  : DIAGNOSTIC_START:
      :     VAL { PASS_COUNTER := ZEROS },
      :     GOTO DIAGNOSTIC_DRIVER.START,
      : ;
      :
0101  : END_DIAGNOSTIC_PASS:
      :     VAL { ALU_BUS := PASS_A (PASS_COUNTER) },
      :     IF VAL (ALU_BUS /= 0) THEN USUALLY GOTO NOT_FIRST_PASS,
      : ;
0102  : COMPLETED_FULL_TEST:
      :     -- Halt at end of first pass --
      :     HALT,
      : ;
      :
0103  : NOT_FIRST_PASS:
      :     VAL { PASS_COUNTER := INC_A (PASS_COUNTER) },
      :     GOTO DIAGNOSTIC_DRIVER.START,
      : ;
```

```
     : package DIAGNOSTIC_DRIVER is

0300 : START:  pragma FIXED_CS_ADDRESS = 16#0300#;
     :
     :     DISABLE (MICRO_EVENTS),
     :     TYP { TYP_BUS := 16#FFFF_FFFF_FFFF_FFFF# },
     :     VAL { ADDRESS_BUS := PASS_A (16#FFFF_FFFF_FFFF_FFFF#) },
     :     RESTORE_MAR_REFRESH,
     : %IF CPU_KIND.MODEL_200_CPU
     :     IOC { SET (CPU_BUSY_LIGHT) },
     : %END
     : ;
     :
0301 : SCAVANGER_INIT:
                .
                .
                .
0323 :
     :     GOTO START_VAL_TEST,
     : ;
     :
030D : START_VAL_TEST:
     :
     :     ACKNOWLEDGE_REFRESH,
     :     CALL VAL_TEST.VAL_TEST,
     : ;
030E :     GOTO START_TYP_TEST,
     : ;
     :
030F : START_TYP_TEST:
     :     ACKNOWLEDGE_REFRESH,
     :     CALL TYP_TEST.TYP_TEST,
     : ;
```

```
     : --TITLE VAL_TEST  -  Value Board diagnostic
     :
     : WITH DIAG_REGS;
     :
     : PACKAGE BODY VAL_TEST IS
     :
     :     RESULT              RENAMES DIAG_REGS.RESULT;
     :     BAD_BITS            RENAMES DIAG_REGS.BAD_BITS;
     :     PATTERN_1           RENAMES DIAG_REGS.PATTERN_1;
     :     PATTERN_2           RENAMES DIAG_REGS.PATTERN_2;
     :     EXPECTED            RENAMES DIAG_REGS.VGP_5;
     :     TEMP_1              RENAMES DIAG_REGS.VGP_5;
     :     TEMP_2              RENAMES DIAG_REGS.VGP_6;
     :     TEMP_3              RENAMES DIAG_REGS.VGP_7;
     :
     : code
     :
0400 : VAL_TEST:       pragma FIXED_CS_ADDRESS = 16#0400#;
     :
     : --
     : --  Test the FALSE condition
     : --
     : COND_FALSE:
     :     IF VAL (FALSE) THEN CALL COND_ERROR,
     : ;
0401 :     IF NOT VAL (FALSE) THEN GOTO COND_FALSE_A,
     : ;
0402 :     CALL COND_ERROR,
     : ;
0403 : COND_FALSE_A:
     :
     :
     : --
     : --  Test the TRUE condition
     : --
     : COND_TRUE:
     :     IF VAL (TRUE) THEN GOTO COND_TRUE_A,
     : ;
0404 :     CALL COND_ERROR,
     : ;
0405 : COND_TRUE_A:
     :     IF NOT VAL (TRUE) THEN CALL COND_ERROR,
     : ;
                    .
                    .
                    .
0926 :     VAL { VAL_BUS := 16#A5A5A5A5A5A5A5A5# },
     :     LOAD_WDR,
     : ;
0927 :     VAL { RESULT := WDR },
     : ;
0928 :     VAL { BAD_BITS := RESULT XOR 16#A5A5A5A5A5A5A5A5# },
     :     IF VAL (RESULT /= 16#A5A5A5A5A5A5A5A5#) THEN
     :                       RARELY CALL CMUX_WDR_ERROR,
     : ;
0929 :     VAL { VAL_BUS := 16#36C936C936C936C9# },
     :     LOAD_WDR,
     : ;
092A :     VAL { RESULT := WDR },
```

```
     : ;
092B :      VAL { BAD_BITS := RESULT XOR 16#36C936C936C936C9# },
     :      IF VAL (RESULT /= 16#36C936C936C936C9#) THEN
     :                      RARELY CALL CMUX_WDR_ERROR,
     : ;

                        .
                        .
                        .

0ADB : COND_ERROR:
     :      DISABLE(MICRO_EVENTS),
     :      HALT,
     : ;
0ADC :      RETURN,
     : ;

                  .
                  .
                  .

0AF1 : CMUX_WDR_ERROR:
     :      DISABLE(MICRO_EVENTS),
     :      HALT,
     : ;
0AF2 :      RETURN,
     : ;
```

Microdiagnostic Listings

```
!!Sunol!R1000.Udiag.Model_200.Listings
---------------------------------------
  Class_Test
  Csa_Test
  Diagnostic_Driver
  Diag_Regs
  Events
  Fiu_Test
  M200_Diag_Map
  Mem_Test
  Mem_Test_Two
  Privacy_Test
  Rf_Test
  Seq_Test
  Sys_Ioc_Test
  Typ_Test
  Val_Test
```

General layout of Microdiagnostic in Control Store

```
0200 - 022D EVENTS
0300 - 0325 DIAGNOSTIC_DRIVER
0400 - 0B08 VAL_TEST
0C00 - 1190 TYP_TEST
1300 - 169F CLASS_TEST
1800 - 1900 PRIVACY_TEST
1A00 - 1C23 FIU_TEST
1E00 - 22DD SEQ_TEST
2400 - 2734 MEM_TEST
2800 - 29F7 MEM_TEST_TWO
2B00 - 2C32 CSA_TEST
2E00 - 2ECE SYS_IOC_TEST
3000 - 32AA RF_TEST
```

Tests for VAL BOARD (may also fail due to problems on SEQUENCER)

VAL_TEST 0400-0B08

COND_ERROR	0ADB	-- possible sequencer
REG_FILE_ERROR	0ADD	
LOGICAL_ALU_ERROR	0ADF	
ARITH_ALU_ERROR	0AE1	
COND_ALU_ERROR	0AE3	
SPLIT_ALU_ERROR	0AE5	
A_PORT_ZERO_ERROR	0AE7	
LOOP_ERROR	0AE9	
CMUX_PASS_ERROR	0AEB	
CMUX_LEFT_ERROR	0AED	
CMUX_RIGHT_ERROR	0AEF	
CMUX_WDR_ERROR	0AF1	
CMUX_FIU_ERROR	0AF3	
FIU_HIGH_ERROR	0AF5	
FIU_LOW_ERROR	0AF7	
FIU_COND_ERROR	0AF9	
ZERO_COUNT_ERROR	0AFB	
ZERO_COND_ERROR	0AFD	
REG_A_B_ERROR	0AFF	
MULT_ERROR	0B01	
DIV_ERROR	0B03	
VAL_BUS_ERROR	0B05	
IMMEDIATE_OP_ERROR	0B07	

Tests for TYP BOARD (may also fail due to problems on SEQUENCER)

TYP_TEST 0C00-1190

COND_ERROR	1165	-- possible sequencer
REG_FILE_ERROR	1167	
LOGICAL_ALU_ERROR	1169	
ARITH_ALU_ERROR	116B	
COND_ALU_ERROR	116D	
SPLIT_ALU_ERROR	116F	
LOOP_ERROR	1171	
CMUX_PASS_ERROR	1173	
CMUX_WDR_ERROR	1175	
CMUX_FIU_ERROR	1177	
ZERO_COND_ERROR	1179	
BIT_21_ERROR	117B	
BIT_32_ERROR	117D	
BIT_33_ERROR	117F	
BIT_34_ERROR	1181	
BIT_35_ERROR	1183	
BIT_36_ERROR	1185	
BIT_33_34_36_ERROR	1187	
CARRY_IN_ERROR	1189	
FIU_HIGH_ERROR	118B	
FIU_LOW_ERROR	118D	
REG_A_B_ERROR	118F	

CLASS_TEST 1300-169F

A_EQ_L_FAILED	1696
B_EQ_L_FAILED	1698
A_EQ_B_FAILED	169A
CLASS_EVENT_FAILED	169C
OF_KIND_ERROR	169E

PRIVACY_TEST 1800-1900

PRIVACY_A_FAILED	18F1
PRIVACY_B_FAILED	18F3
PRIVACY_NAMES_EQ_FAILED	18F5
PRIVACY_A_B_FAILED	18F7
PRIVACY_EQ_FAILED	18F9
PRIVACY_STRUCTURE_FAILED	18FB
PRIVACY_PATHS_EQ_FAILED	18FD
PRIVACY_CHECKER_ERROR	18FF

Tests for FIU BOARD

FIU_TEST 1A00-1C23

OFFSET_ERROR	1BFE	
LENGTH_ERROR	1C00	
XWORD_ERROR	1C02	
MAR_WORD_EQ_ZERO_ERROR	1C04	
VAR_ERROR	1C06	
TAR_ERROR	1C08	
MDR_ERROR	1C0A	
VI_MUX_FIU_ERROR	1C0C	
MERGE_V_MUX_FIU_ERROR	1C0E	
TI_MUX_FIU_ERROR	1C10	
FILL_BIT_ERROR	1C12	
MERGER_EXTRACT_ERROR	1C14	
MERGER_INSERT_V_ERROR	1C16	
MERGER_INSERT_T_ERROR	1C18	
MERGER_INSERT_ERROR	1C1A	
ROTATE_ERROR	1C1C	
ROTATOR_ERROR	1C1E	
EXTRACT_ERROR	1C20	
INSERT_ERROR	1C22	

Tests for SEQUENCER BOARD (may also fail due to problems on TYP OR FIU)

SEQ_TEST 1E00-22DD

FAILED_DISPATCH	2061	
BAD_OPTIMIZATION_ERROR	2066	
BAD_USUALLY_DISPATCH	20F6	
BREAK_ERROR	226B	
COND_ERROR	226C	
UNSUCCESSFUL_USUALLY_CASE	20FA	
WRONG_CASE_VALUE	20FE	-- possible FIU
BREAK_ERROR	226B	
COND_ERROR	226C	
BAD_MACRO_PC	226E	
BAD_LEX_VAL	2270	
BAD_MACRO_PC_REF	2272	
BAD_MACRO_PC_SEG	2274	
BAD_CODE_MUX	2276	
BAD_MACRO_PC_INC	2278	
BAD_MACRO_PC_ADD	227A	
BAD_RETURN_PC	227C	
BAD_RETURN_PC_REF	227E	
BAD_RETURN_PC_SEG	2280	
BAD_RETURN_PC_LOAD	2282	
BAD_CONTROL_PRED	2284	
BAD_CONTROL_TOP	2286	
BAD_SEQUENCER_BUS	2288	
BAD2_CONTROL_PRED	228A	-- possible FIU or TYP
BAD2_CONTROL_TOP	228C	-- possible FIU or TYP
BAD_CONTROL_TOP_INC	228E	-- possible FIU or TYP
BAD_CONTROL_TOP_DEC	2290	-- possible FIU or TYP
BAD_RESOLVE	2292	
VALIDATE_BAD	2294	
BAD_RESOLVE2	2296	

```
BAD_RESOLVE_ADDRESS                 2298
VALIDATE_BAD2                       229A
BAD_INVALIDATE                      229C
BAD_CURRENT_INSTR                   229E
BAD_CURRENT_INSTR1                  22A0
BAD_IBUFFER                         22A2
BAD_IBUFFER2                        22A4
BAD_IBUFFER3                        22A6
BAD_SAVE_OFFSET                     22A8
BAD_CURRENT_NAME                    22AA
BAD2_SAVE_OFFSET                    22AC
BAD2_CURRENT_NAME                   22AE
BAD_NUMBER_VALID                    22B0
BAD_NUMBER_VALID2                   22B2
BAD_NUMBER_VALID3                   22B4
BAD_NUMBER_VALID4                   22B6
BAD_DISPATCH_ZERO                   22B8
BAD_DISPATCH_ONE                    22BA
PC_UPDATE_ERROR                     22BC
IBUFF_FILL_ERROR                    22BE
DISPATCH_INDEX_ERROR                22C0
IFILL_ERROR                         22C2
BAD_CONDITIONAL_LOAD_IBUFF          22C4
BAD_CONDITIONAL_LOAD_IBUFF2         22C6
TOS_VALIDATE_FAILED                 22C8
TOS_LATCH_DIDNT_HOLD                22CA
DISPATCH_DIDNT_INVALIDATE           22CC
DIDNT_TAKE_MICRO_EVENT              22CE
FIELD_NUMBER_TEST_FAILED            22D0
NVE_ERROR                           22D2    -- possible FIU or TYP
NFREE_ERROR                         22D4    -- possible FIU or TYP
RES_REF_ERROR                       22D6
RES_REF_ERROR_1                     22D7
TOS_OP_ERROR                        22D9
TOS_OP_ERROR_ONE                    22DA
MICRO_STACK_ERROR                   22DC
```

Tests for MEMORY BOARDS (may also fail due to problems on FIU)

MEM_TEST 2400-2734	
INIT_ERROR	2727
MAR_ERROR	2729
MEM_BOARDS_NOT_ALL_SAME_SIZE	272B
TAG_STORE_DATA_ERROR	272D
HASH_ERROR	272F
TAG_STORE_COMPARE_ERROR	2731
RAM_PLANE_DATA_ERROR	2733

MEM_TEST_TWO 2800-29F7	
FLAG_ERROR	29D8
E_ABORT_ERROR	29DA
READ_ERROR	29DC
PAGE_ERROR	29DE
COND_ERROR	29E0
FRAME_ERROR	29E2
PAGE_XING_ERROR	29E4
INIT_ERROR	29E6
LRU_MRU_ERROR	29E8
COND_CONT_ERROR	29EA
LOGICAL_ERROR	29EC
TVR_ERROR	29EE
VERIFY_ERROR	29F0
ECC_EVENT_NO_TAKEN	29F2
ECC_ERROR	29F4
AVAIL_ERROR	29F6

Tests for CSA which is made up of logic on multiple boards
Failures could be caused by (TYP, FIU, VAL, SEQUENCER)

CSA_TEST 2B00-2C32	
ERROR_HALT	2C19
CSA_HIT_ERROR	2C1B
OUT_OF_RANGE_ERROR	2C1D
CSA_INIT_ERROR	2C1F
CSA_READ_ERROR	2C21
CSA_WRITE_ERROR	2C23
MEM_ERROR	2C25
BAD_HINT_ERROR	2C27
CANNOT_ALIGN_BOT_ERROR	2C29
TYP_VAL_BOT_UNEQUAL_ERROR	2C2B
DEC_CSA_BOT_ERROR	2C2D
INC_CSA_BOT_ERROR	2C2F
CSA_BOTM1_ERROR	2C31

Tests for IOC BOARD

SYS_IOC_TEST 2E00-2ECE	
TIMER_ERROR	2EC7
TIMER_COUNT_ERROR	2EC9
SLICE_MACRO_EVENT_ERROR	2ECB
GP_MACRO_EVENT_ERROR	2ECD

Test for Register file

This test is normally not run since the values used in the previous test come out of the register file area, and running this test will destroy them. The RF_TEST can only be run from EXPMON.

RF_TEST 3000-32AA

VAL_INC_FAILED	3241
TYP_COUNTER_ERROR	3243
TYP_INC_FAILED	3245
VAL_ALU_ZERO_CHECK_FAILED	3247
VAL_ALU_ONES_CHECK_FAILED	3249
VAL_REG_COMPARE_FAILED	324B
VAL_REG_INC_FAILED	324D
VAL_INC_A_FAILED	324F
VAL_START_BIT64_ERROR	3251
VAL_BIT64_ERROR	3253
VAL_FINAL_BIT64_ERROR	3255
VAL_START_ALU_NE_0_ERROR	3257
VAL_ALU_NE_0_ERROR	3259
VAL_FINAL_ALU_NE_0_ERROR	325B
TYP_ALU_ZERO_CHECK_FAILED	325D
TYP_ALU_ONES_CHECK_FAILED	325F
TYP_REG_COMPARE_FAILED	3261
TYP_REG_INC_FAILED	3263
TYP_INC_A_FAILED	3265
TYP_START_BIT64_ERROR	3267
TYP_BIT64_ERROR	3269
TYP_FINAL_BIT64_ERROR	326B
TYP_START_ALU_NE_0_ERROR	326D
TYP_ALU_NE_0_ERROR	326F
TYP_FINAL_ALU_NE_0_ERROR	3271
TYP_A_SIDE_ZERO_ERROR	3273
TYP_B_SIDE_ZERO_ERROR	3275
TYP_A_B_ZERO_ERROR	3277
VAL_A_SIDE_ZERO_ERROR	3279
VAL_B_SIDE_ZERO_ERROR	327B
VAL_A_B_ZERO_ERROR	327D
TYP_A_SIDE_ONES_ERROR	327F
TYP_B_SIDE_ONES_ERROR	3281
TYP_A_B_ONES_ERROR	3283
VAL_A_SIDE_ONES_ERROR	3285
VAL_B_SIDE_ONES_ERROR	3287
VAL_A_B_ONES_ERROR	3289
TYP_COUNTER_CONTENTS_ERROR	328B
TYP_COMPARE_ERROR	328D
VAL_COUNTER_CONTENTS_ERROR	328F
VAL_COMPARE_ERROR	3291
VAL_REG_BAD_LOAD	3293
VAL_NIBBLE_ERROR	3295
TYP_REG_BAD_LOAD	3297
TYP_NIBBLE_ERROR	3299
VAL_A_SIDE_ADDR_ERROR	329B
VAL_A_SIDE_FRAME_ERROR	329D
VAL_B_SIDE_ADDR_ERROR	329F
VAL_B_SIDE_FRAME_ERROR	32A1
TYP_A_SIDE_ADDR_ERROR	32A3
TYP_A_SIDE_FRAME_ERROR	32A5
TYP_B_SIDE_ADDR_ERROR	32A7
TYP_B_SIDE_FRAME_ERROR	32A9

RUNNING THE MICROCODE DIAGNOSTIC

```
CLI/CRASH MENU - options are:
  1 => enter CLI
  2 => make a CRASHDUMP tape
  3 => display CRASH INFO
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [make a CRASHDUMP tape] : 1
CLI> x fru
Rational R1000 FRU diagnostic driver

Main FRU menu
    1 => Display cluster information
    2 => Execute confidence test (microdiagnostic)
    3 => Execute diagnostics
    4 => Execute PM tests (not implemented yet)
    5 => Margin cluster clocks/power
    6 => Specify test options
    7 => Repair assistance (model 100 CPU power control)
    8 => Initialize processor state
    0 => Exit
Please enter option : 8

Main FRU menu
    1 => Display cluster information
    2 => Execute confidence test (microdiagnostic)
    3 => Execute diagnostics
    4 => Execute PM tests (not implemented yet)
    5 => Margin cluster clocks/power
    6 => Specify test options
    7 => Repair assistance (model 100 CPU power control)
    8 => Initialize processor state
    0 => Exit
Please enter option : 2
preparing to run the Confidence Test (uDIAG)
The long version stress tests the DRAMs but runs 2 minutes longer'
Do you want to run the long version [N] ? n
  Loading from file DIAG.M200_UCODE  bound on November 15, 1989  13:02:00
  Loading Register Files and Dispatch Rams .... [OK]
  Loading Control Store ............ [OK]
the Confidence test (uDIAG) passed
```

--- FAILURE EXAMPLE 1 ---

```
Main FRU menu
    1 => Display cluster information
    2 => Execute confidence test (microdiagnostic)
    3 => Execute diagnostics
    4 => Execute PM tests (not implemented yet)
    5 => Margin cluster clocks/power
    6 => Specify test options
    7 => Repair assistance (model 100 CPU power control)
    8 => Initialize processor state
    0 => Exit
Please enter option : 2
preparing to run the Confidence Test (uDIAG)
The long version stress tests the DRAMs but runs 2 minutes longer
```

```
Do you want to run the long version [N] ? n
  Loading from file DIAG.M200_UCODE  bound on November 15, 1989  13:02:00
  Loading Register Files and Dispatch Rams .... [OK]
  Loading Control Store ............ [OK]
detected a MACHINE_CHECK, therefore Running FRU EMBALM
uaddr trace:   4=091C  3=091D  2=091E  1=091F   --ucode M207_39
VAL board WDR parity error -
uDIAG failed: a MACHINE_CHECK occured


     --- FAILURE EXAMPLE 2 ---
Main FRU menu
    1 => Display cluster information
    2 => Execute confidence test (microdiagnostic)
    3 => Execute diagnostics
    4 => Execute PM tests (not implemented yet)
    5 => Margin cluster clocks/power
    6 => Specify test options
    7 => Repair assistance (model 100 CPU power control)
    8 => Initialize processor state
    0 => Exit
Please enter option : 2
preparing to run the Confidence Test (uDIAG)
The long version stress tests the DRAMs but runs 2 minutes longer
Do you want to run the long version [N] ? n
  Loading from file DIAG.M200_UCODE  bound on November 15, 1989  13:02:00
  Loading Register Files and Dispatch Rams .... [OK]
  Loading Control Store ............ [OK]
*** Confidence test (uDIAG) failed an error halt occured at uaddr 0ADC,
```

4. Peripheral Testing

From the CLI there are several programs available for checking out the peripheral devices.

CHECKDISK – Disk Surface analysis

DISKX – Disk exerciser

TAPEX – TAPE exerciser

CHECKDISK

The CHECKDISK program performs a non-destructive, fast, read-only surface analysis of a formated, labeled disk. CHECKDISK will avoid previously detected bad blocks on a labeled disk, reporting only errors which are not known.

Run the CHECKDISK program by type:

```
CLI> x checkdisk
```

The CHECKDISK program will ask for the disk unit number to check. The disk must have been previously formatted and labeled with the RECOVERY program. Next the program will ask for the number of passes desired. Normally a single pass is sufficient but for internal or testing purposes several passes may be desired. The program will then ask if you want error information displayed for all defects or only previously undetected defects. Normally only previously undetected error information is desired but at times all information is useful.

Once the program begins it will display the current cylinder number constantly and print a pass counter at the conclusion of each pass.

DISKX

The DISKX program is a DFS based disk exerciser. It is intended to be used for quick disk subsystem integrity testing as well as long term reliability testing. To invoke the program type:

```
CLI> x diskx
```

The program will size the system and query the operator about testing each existing disk. After these questions the exerciser will begin testing. Once testing has begun the exerciser may be stopped by typing control-G. Any other character will cause the program to display status information about each drive. This status information includes the time the test was started, the current time, total number of bytes transferred, total hard errors, total soft errors, and current head location.

The exerciser does transfers in the following manner.

- write random data to a random block in the diagnostic region.
- check the data just written.
- read a random block from anywhere on the disk.

These three steps are repeated for each unit with all units running in parallel until the test is stopped.

TAPEX

The TAPEX program is a DFS based tape exerciser. It is intended to be used for quick tape subsystem integrity testing as well as long term reliability testing. To invoke the program type:

```
CLI> x tapex
```

The program will size the system and query the operator about testing each existing tape drive which is on-line. After these questions the exerciser will begin testing. Once testing has begun the exerciser may be stopped by typing control-G. Any other character will cause the program to display status information about each drive. This status information includes the time the test was started, the current time, total number of bytes transferred, total hard errors, total soft errors, and data errors.

The exerciser does transfers in the following manner.

- Select a random data pattern and write a record of random length. This step is repeated a random number of times between 16 and 63.
- Backspace a random number of records between 1 and the number of records just written.
- Read and check the data of the records just backspaced over.

These three steps are repeated for each unit with all units running in parallel until the test is stopped.

Dealing with Problem Situations

In this section we will review general steps to be followed for the following situation

- Hung System
- Hardware/Software Crash
- Disk Full / Suspend System Threshold

HUNG System

Procedures for System Hang Condition

To perform preliminary diagnosis and provide the maximum amount of information for diagnosis of a system, the following procedures should be performed prior to crashing the system and possibly taking a crashdump. Whenever a crashdump tape is taken, it is very important to capture as much information about the crashdump as possible. The following procedures describe the minimal amount of information necessary based on the type of the system hang.

No System Console Response

The system console might be flow controlled, which will make it appear as if the system is hung.

1. Does the console respond to the BREAK key with the "Please enter 0/1/2" message? If not, try typing "2", followed by a <CR>. If the console now responds to the BREAK key, then the console was flow controlled by the IOA/C waiting for the "0/1/2" response. If the console still does not respond, then there is a problem with the console terminal, the connection to the IOA/C, the IOA/C itself, or another piece of hardware. Verify that the terminal is functioning correctly (power the terminal OFF, then ON to initiate self testing). In the event that no response can be obtained from the R1000, the system will likely require a hard, "white-button" reset. In this event all information will have been cleared and a crashdump will be of no use.
2. Does the console switch banners in response to ^Z? If not, try typing <CR>. If there is still no response, then type ^Q. If the console now responds to ^Z or <CR>, then the console was flow controlled by R1000 or IOP/C software. If the console still does not respond, then crash the system (using the BREAK key).

System Console Responds

Having reached this point, it has been determined that the system console will respond. If the prompt on the console is "CLI>" or "Enter configuration to boot..", then the machine has crashed, and the cause of the crash was displayed in previous output to the console. If this output is still visible, examine it to determine the cause of the crash and what was recommended as appropriate action.

It is important to note that:

1. Both interpreters (Kernel and EEDB) can present the "Kernel:" prompt.
2. By typing <CR> and using ^Z one can cycle through the prompts to determine which interpreter is displaying the "kernel:" prompt (indicated in the banner information)
3. To get the "Kernel:" prompt via the EEDB interface, type "kernel" at the EEDB prompt.
4. It is always better to use the kernel prompt under the EEDB banner. If the EEDB becomes hung, the non-faulting Kernel interpreter is still available.

Can the Kernel or EEDB command interpreters be reached? If not, gather the following information:

1. The output produced by BREAK-2.
2. The banners displayed by ^Z.
3. The prompts that are displayed on the console (use ^Z)

then crash the system (using the BREAK key).

Having reached this point, it has been determined that the R1000 is still running and that the Kernel/EEDB interpreters respond.

1. Do "Show_Disk_Summary" at the Kernel prompt. Does it show unrecoverable disk errors? If so, this is most likely the reason for the hang.
2. Does "Show_Disk_Summary" show IO in progress? If so, execute the command several times. If the successive displays show no change in the total read/writes columns, yet the display shows disk IO in progress, there is an IO hang. Execute a "Show_Task_States", record the information.
3. Do "Show_Volume_Summary". If any volume shows 0 unused capacity, then the system has reached the suspend system GC threshold. This can be verified by noting that no waiters exist for the suspend threshold for that volume. Reboot following the procedures in the System Manager's Guide for recovering when the Suspend_System threshold has been reached.
4. Do a "Jobs" (threshold 1) command. Do executions of this command show activity in the CPU and D/S categories? There should be either virtually no activity, or some job(s) consuming virtually all cycles.
5. If the problem is too much activity, wait 5 or 10 minutes before proceeding. If the activity is in user jobs, then disable the user job. Determine what the user job is doing and take the appropriate action. If the activity is in job 4 or 5, record several executions of the "Show_Task_States" (Cache).

Having reached this point, the R1000 is still running, the Kernel/EEDB interpreter respond, and the system appears to be idle, yet it also appears to be hung.

1. Use "Show_Error_Log" and search for anything out of the ordinary.

Having reached this point, the R1000 is still running, the Kernel/EEDB interpreter respond, and the system appears to be idle, yet it also appears to be hung.

1. Use "Show_Error_Log" and search for anything out of the ordinary.
2. Can users login? If not, try more than one connection. If telnet is being used, try a serial port.
3. Are users logged in but getting no response? Are all users having this problem?

It is possible for a user's session to become hung, which doesn't imply that all sessions are hung. If not all user's sessions are hung, examine the session error logs (in !Machine.Error_Logs) for the hung sessions. Also do a "Show_Task_States" (Cache) and see if any task is in an "Environment_Debugging_Wait". If so, execute the Show_Tasks procedure in the System_Maintenance subsystem, using the task ID of the task shown to be in the "Environment_Debugging_Wait". A determination should then be made as to when an appropriate time would be to reboot the system to clear the hung session.

Hardware/Software Crash

If you find that the system has crashed, the following procedure may help in isolating the cause. The procedure performs two fundamental operations:

- capturing system crash state
- and analyzing system hardware

1) If the Operator Mode keyswitch on the R1000's control panel is not in the Interactive position, turn it to that position.
2) Write down the context of the crash.
    a. Write down the machine check message, PC, and error code.
    b. Write down any other messages or clues about why the R1000 crashed.
    c. How long had the R1000 been running since the last boot?
    d. Was the R1000 still booting when it crashed? Why was it booting?
    e. Was the Environment up at the time?
    f. Was there anything abnormal about the way the Environment was functioning?
    g. What were users doing? Any unusual activities?
    h. Were any daemons (snapshot, daily, or weekly) running?

i. Was a tape drive in use?

j. Have there been any hardware, firmware, or software changes to this R1000 within the past week?

k. Have there been any problems with or modifications of the network in the past week?

l. What was the temperature in the computer room at the time of the crash

m. Have there been any problems with the air conditioning recently?

n. Are there other computers (R1000s or not) in the same building? Did any of the other computers crash or have power or environmental problems at the same time?

3) If the operator console has either of these prompts,

Do you want to run FRUs?

or

Do you want to take a crash dump?

Answer N (No) and answer the other prompts to go to the CLI.

4) CLI> x expmon

a. EM> sm

b. Write down the results.

5) EM> trace

a. Write down all two or three columns of the last ten entries numbered 9,8,7...0.

b. Some interesting halt addresses are
0200_HAVE_SYSTEM_ERROR
0201_HAVE_IOP_HARDWARE_ERROR
0202_HAVE_IOP_SOFTWARE_ERROR
0204_HAVE_MULTI_BIT_MEMORY_ERROR
0205_HAVE_SYSBUS_HARDWARE_ERROR
0208_HAVE_SYSTEM_SHUTDOWN

6) CLI> x crashdump

a. Make a crash dump tape. On an 8-mm tape drive, any tape at least 15 meters long (15 minutes video playing time) will be sufficient. The tape should be made because it might be possible to analyze it at your site.

7) CLI> fptest -- Test the foreplane.

8) CLI> x fru -- Run FRU diagnostic tests.

a. Select option 8 => Initialize processor state

b. Select option 3 => Execute diagnostics

c. Select option 2 => Run all tests

d. Request maximum phase 3 for best results. Phase 3 will take about 45 minutes. If you need to reboot sooner, request maximum phase 2 (which will take about 15 minutes) or phase 1 (about 3 minutes).

e. Write down the results. If a test fails, write down its name and any corresponding messages.

f. Select option 0 => Exit

g. Select option 8 => Initialize processor state

h. Select option 0 => Exit

9) Run the functional diagnostics.

a. CLI> x fru -- Run FRU diagnostic tests

b. Select option 8 => Initialize processor state

c. Select option 2 => Execute Confidence tests

d. Write down the results of the Udiag. Will take about a minute to run.

10) If the functional diagnostics did not end with a success message

a. CLI> x expmon

b. EM> sm

c. Use the "trace" command to list the last addresses. Use this information to map to the area of the diag that was under execution at the time of the halt.

11) CLI> x bootinfo -- Get the current microcode version from the output.

12) CLI> x novram -- Display (NEVER modify!!!) data for all boards and write down the output. Then exit.

13) CLI> x log -- Get new messages of all categories and write down the results. Do not update the log header.

14) CLI> x diskx

a. Run the disk exerciser on all disks for at least five minutes.

b. Press the space bar to cause the latest output to be displayed. The output will appear something like this:

```
u0  bytes=> 1246208    soft=> 0     hard=> 0     C=> 705  T=> 25  S=> 6
u1  bytes=> 1203200    soft=> 0     hard=> 0     C=> 73   T=> 14  S=> 8
u2  bytes=> 1206272    soft=> 0     hard=> 0     C=> 611  T=> 3   S=> 24
u3  bytes=> 1243208    soft=> 0     hard=> 0     C=> 634  T=> 11  S=> 10
```

c. Enter control-G to stop DISKX. This is a read-and-write test, but it only uses a small test area of each disk, so it will not harm user data on the disks.

d. Write down the results.

15) If you suspect disk problems, run CHECKDISK on all disks or on each disk you suspect. Note: CHECKDISK may take up to 30 minutes per disk per pass! Request that only new errors be reported. Write down the HDA numbers if you don't already have them recorded anywhere. This is a read-only test and will not harm user data on the disks. Run at least one pass of surface analysis, more if you prefer.

Note: Commands typed-ahead are not discarded.

a. CLI> x checkdisk

b. Write down the results.

16) CLI> boot -- Try rebooting the R1000.

a. Write down the results.

b. If you successfully reboot, return the Operator Mode key-switch to its original position.

Disk Full / Suspend System Threshold

Hangs caused by the Suspend System Threshold being crossed were mentioned under possible hangs. However this issue warrants additional coverage.

The system_system threshold level is the result of the disk volume filling with too much data. (Both true data and garbage). Once in this situation the steps outlined in the systems management guide should be followed to clear the problem.

Users of the R1000 system should be warned that if it is noted that as system is approaching the Suspend level, that they should not attempt to address the issue by deleting large files/objects. This will only result in the generation of addition garbage, making the situation worse. Cleanup should be delayed until after the system has had a chance to clean garbage.

Refer to the System Manager's Guide for information on how to recover from the Suspend_System Threshold or Stop_Jobs threashold, under the "Threshold Recovery" tab.

R1000 Software Install

On an R1000 system, the install of software and creation of a bare machine has changed as the machine moved to end-of-life.

When the system was under active development, various forms of media were made available for the installation of components and products. Now that the project has gone end-of-life, much of that technology is unavailable.

R1000 Media Types

1) DFS Backup

   This tape is generated from the DFS level and contains all to contents of the DFS area. This tape is loaded onto a system by running the "recovery" program. The recovery program is on the DFS tape, and can be launched by setting the R1000 to boot from tape.

2) AK tape

   The AK tape, otherwise known as the Kernel Tape, would be used to load at the DFS level the various Kernel level code segments. An AK tape would be the method to load updated code segments into the DFS level.

3) AE tape

   The AE tape would be used to load code segments and configuration information into the Environment/EEDB layer. As with the AK tape, this tape was used to load updates, or to reload files when corruption was suspected.

4) Tools tape

   The Tools tape would be used to install new layered products, or updates, into the Delta environment. In general the tapes were R1000 archives that would be restored onto the target machine.

5) Environment tape (ENV tape)

   The Environment tape would be used to install an update of the Delta Environment.

6) System Backup

   The system backups are made from a running system and are a means of recovery should there be a disk failure, or other loss of data on a system.

   There is a sub category of the system backup which was referred to as a "virgin" backup. This was a backup of a bare system, having just the environment but no user data. These backups would be used in the process of setting up a new machine.

With the end of development of the R1000 systems, the AE and AK tapes ceased to be generated. Without new development there were no fresh code segments to install.

Over the years the Tools and Environment tapes have also been discarded as the system was no longer supported. However some sites may still have copies for products delivered to their locations.

Because of the low likelihood that Tools tapes are available, taking a clean system and loading it up with a defined set of add-on products would be a difficult, if not impossible, effort.

At this point in time, the most likely path for loading a useable environment onto a new/clean machine would be to recover it from backups of an existing system with the components of interest. However if an individual did have access to the tools tapes needed they could possibly take the approach of starting with a virgin system. This would require a virgin backup for the Delta release version of interest.

Information on recovering a system from a DFS/Environment backup is contained in the System Manager's Guide, Chapter 7. (page 150).

Installation Procedure for Rational X Interface Release10_5_2

RXI allows users a method of accessing the Rational Environment from a workstation using X Windows or a X Terminal.

The installation materials for this release are:

- This Installation Procedure
- RXI Release10_5_2 Release Note
- RXI Release10_5_2 Tape

Impact of the Release

Installation of this product will require that the R1000 be rebooted in order for the new keymaps to take affect.

Installation Overview

There are two general classes of RXI-based terminal types and the steps in their installation may differ somewhat.

- "Real" workstations. For example, you have a Sun workstation on your desk and that is the workstation that you will be using as an RXI "terminal" when working with an R1000. You would be using the "real" terminal type called **`Xsun4_X11R3`** (or one of the other Sun variations).
- "Parasite" terminals. For example, you have an NCD X Terminal on your desk that keeps its font files and such on the file-system of a nearby DEC Vax running VMS. To use the NCD as your RXI terminal you would use the "parasite" terminal type called **`Xncd_Vms5_1_Multinet`** (or one of the other NCD/VMS variations).

The general installation steps for the mythical XYZ terminal type are:

- Execute the **`Rxi_Install.Machine_Editor_Data_Files`** procedure to install the various XYZ keymap files on your R1000. This automatically enables any product-specific commands contained in the new keymap(s). Commands are enabled if the pertinent product is authorized for this R1000.

This step also creates XYZ entries in the **!Machine.Editor_Data.[Terminal_Types, Terminal_Recognition]** files.

This step is performed for all "real" and all "parasite" terminal types that will be used to talk to the R1000. It installs the R1000 key map for this terminal type and it tells the R1000 how to perform auto-recognition of the new terminal type when a user logs into the R1000.

This step is performed once for each rxi terminal type that your site will be using, real or parasite.

- Execute the **Rxi_Install.Workstation_Files** procedure to transfer a full set of RXI source and executable files to the workstation. Follow the instructions attached to the installation procedure down below when performing this step. This step copies, creates, and/or installs the rxi program upon the workstation.

  This step is usually performed once for a particular file system or file server. RXI only needs to be installed once on a file system or file server. It is installed with a system-wide default terminal type corresponding to a single R1000 terminal type. Users with other R1000 terminal types then use the **-rcg** switch (or the ***recognition:** resource) when invoking RXI.

- Reboot the R1000.

The majority of the steps which must be performed for this installation utilize the procedure **Do_Step** (included as part of the release tape and located in **!Commands**) to execute the required sequence of commands which implement the step. This procedure **must always** be executed from a command window in the release library

```
!Machine.Release.Archive.X_Interface.Release10_5_2
```

Each step which utilizes procedure **Do_Step** will be of the form

<STEP_NAME> <DESCRIPTION>

where **<STEP_NAME>** is passed as the parameter to procedure **Do_Step** which performs the necessary actions to complete the step, and **<DESCRIPTION>** is a description of what the step does, including any action which you will need to manually perform. For example:

FOO

This step is an example of the format used for steps which are executed using procedure **Do_Step**. To execute step **FOO**, you would go to **!Machine.Release.Archive.X_Interface.Release10_5_2** and in a command window, execute:

```
Do_Step ("FOO");
```

which would result in automatic execution of all commands required to implement this step. If you

needed to take any manual action in addition to executing **Do_Step**, it would be noted as :

❒ Perform some manual check

If any errors occur for this step, you should always fix the problem before proceeding on to the next step. Each step is defined by a fragment of Ada code which is executed by **Program.Run**. These fragments are stored in the **Step** file located in the **Command_Data** library of the release. In the event you want to modify a step, you can use the "PROMPT => <STEP>" form when invoking **Do_Step**. See the spec of procedure **Do_Step** (located in **!Commands**) for more detailed information.

Warning: If you interrupt the execution of ***Do_Step*** *(by using* ***Job.Interrupt*** *such as* ***CONTROL-G****) it is possible that certain interactive commands executed by some steps, such as* ***Common.Definition****, may fail with an exception and display a message such as*

```
Unable to read file due to Constraint_Error (Null Access)
```

In general, this message has no negative impact on the execution or completion of the step, and can be ignored.

Prerequisites

❒ 1. A workstation or file-server with supported versions of X Windows and TCP/IP (see list below). The X Windows and TCP/IP must be installed and operational on the workstation before RXI can be installed.

❒ 2. R1000 with Environment release D_10_20_0 or later.

❒ 3. Verify that the user community has been notified of installation down time.

❒ 4. Review the complete Installation Procedure before proceeding.
Warning: *Be sure to complete ALL of the phases of the installation exactly as written and to follow the installation steps in each phase in a serial manner. Failure to adhere to these instructions may result in the failure of portions of the installation.*

Supported Configurations

The following terminal type configurations are supported in this release:

- **Xnews3_V1_0** - Sun M680x0 with SunOS 4.0,version 1.0 of Sun's X11/NeWS and type 3 keyboard.
- **Xnews3_V1_0_Sparc** - Sun Sparc with SunOS 4.0, version 1.0 of Sun's X11/NeWS and type 3 keyboard.
- **Xnews4_V1_0** - Sun M680x0 with SunOS 4.0, version 1.0 of Sun's X11/NeWS and type 4 keyboard.
- **Xnews4_V1_0_Sparc** - Sun Sparc with SunOS 4.0, version 1.0 of Sun's X11/NeWS and type 4 keyboard.
- **Xsun3_X11r3** - Sun M680x0 with SunOS 4.0, MIT's X11.R3 X Window System and type 3 keyboard.
- **Xsun3_X11r3_Sparc** - Sun Sparc with SunOS 4.0, MIT's X11.R3 Window System and type 3 keyboard.
- **Xsun4_X11r3** - Sun M680x0 with SunOS 4.0, MIT's X11.R3 X Window System and type 4 keyboard.
- **Xsun4_X11r3_Sparc** - Sun Sparc with MIT's X11.R3 Window System and type 4 keyboard.
- **Xrtus_Aix2_1** - IBM PC/RT with AIX 2.2.1, AIX-X Windows 2.1 (X11 update levels to 02.01.0070).
- **Xrtus_Aix2_1b** - IBM PC/RT with AIX 2.2.1, AIX-X Windows 2.1 (X11 update levels above 02.01.0070).
- **Xdecus_Vms5_1_Multinet203** - DEC VAXstation with VMS 5.1 or later, DECwindows, LK201 U.S. Keyboard, Excelan MultiNet TCP/IP release 2.0.3 or later.
- **Xdecus_Vms5_1_Wollongong502** - DEC VAXstation with VMS 5.1 or later, DECwindows, LK201 U.S. Keyboard, Wollongong WIN TCP/IP release 5.0.2 or later.
- **Xncd_Xnews_V1_0** - NCD X Terminal, Sun M680x0 host with SunOS 4.0, X11/NeWS V1.0.
- **Xncd_Xnews_V1_0_Sparc** - NCD X Terminal, Sun Sparc host with SunOS 4.0, X11/NeWS V1.0.
- **Xncd_Sun_X11r3** - NCD X Terminal, Sun M680x0 host with SunOS 4.0, MIT's X11.R3.
- **Xncd_Sun_X11r3_Sparc** - NCD X Terminal, Sun Sparc host with SunOS 4.0, MIT's X11.R3.
- **Xncd_Aix2_1** - NCD X Terminal, IBM PC/RT host with AIX 2.2.1, AIX-X Windows 2.1 (X11 update levels to 02.01.0070).

- **Xncd_Aix2_1b** - NCD X Terminal, IBM PC/RT host with AIX 2.2.1, AIX-X Windows 2.1 (X11 update levels above 02.01.0070).

- **Xncd_Vms5_1_Multinet203** - NCD X Terminal, DEC VAXstation host with VMS 5.1, DECwindows, Excelan Multinet TCP/IP 2.0.3 or later and VAX/VMS software from NCD.

- **Xncd_Vms5_1_Wollongong502** - NCD X Terminal, DEC VAXstation host with VMS 5.1, DECwindows, Wollongong TCP/IP 5.0.2 or later and VAX/VMS software from NCD.

Installation on R1000

❒ 1. Log into the system as a user who is a member of group PRIVILEGED. Make sure you are a member of group Privileged.

```
Operator.Display_Group ("privileged")
```

❒ 2. Load RXI Rev10_5_2 tape onto the tape drive.

❒ 3. Create a command window and restore the tape onto the system by executing:

```
Operator.Enable_Privileges;
if Operator.Privileged_Mode then
    Archive.Restore (Options => "Replace, Promote");
else
    Io.Put_line ("User does not have privileges.");
end;
```

Answer the appropriate tape mount requests on the system console.

This will restore **!Machine.Release.Archive.X_Interface.Release10_5_2** (after about 20 minutes).

❒ 4. Go to the **!Machine.Release.Archive.X_Interface.Release10_5_2** library. The following steps utilize the procedure **Do_Step** described earlier in this document.

❒ 5. **AUTHORIZATION_CHECK** This step will verify that the appropriate products are authorized for this machine. If there are products which need to be authorized, this will be indicated and a command window created with a call to procedure **Product_Authorization.Register**. Modify the appropriate parameters and authorize the products required.

❒ 6. **RELEASE_RESTORE** This step restores the and **Release** archives. When completed (about 20 minutes), a filtered error log is displayed. Examine this log for errors, ignore errors about switches and links.

❒ 7. **RESTORE_NOTES** This step restores the **_PS** and **_LPT** versions of the RXI release note to

```
!Machine.Release.Release_Notes.
```

❒ 8. Traverse to `!Machine.Release.X_Interface.Rev10_5_2`.

❒ 9. Install the keymap files and update the **Terminal_Type** and **Terminal_Recognition** files on the R1000. This step must be repeated for each terminal type you want to use with this R1000. For example, to install the files for a Sun workstation with X11R3 and a type 4 keyboard, enter the following in a command window:

```
Rxi_Install.Machine_Editor_Data_Files
     (Keyboard => Rxi_Install.Xsun4_X11r3,
      Suppress_Rxi_files => False);
```

and press the [PROMOTE] key. If you are installing more than one terminal type on this R1000, you can save time by setting the **Suppress_Rxi_Files** parameter to **True** after this command has been executed once.

Reboot the R1000

The R1000 must be rebooted for the new keymaps to take affect. The best time to reboot the R1000 is after you have transferred the RXI source files to the workstation, this way you can install RXI on the workstation while the R1000 is rebooting.

Use the instructions in the following sections to transfer and install RXI on your workstation. After the files have been transferred to the workstation reboot the R1000. In a Command window, type:

```
Schedule_Shutdown
     (At_Time       => ">> Some Future Time <<",
      Reason        => "Release",
      Explanation   => "Install RXI Rev10_5_2");
```

and press the [PROMOTE] key when you are ready to reboot the R1000.

Test

After the installation is complete, log-on to the R1000 from the workstation and try some function keys and menus to verify RXI is working correctly.

Clean Up

After the installation is complete, you may delete some objects to save some disk space.

- `!Machine.Release.Archive.X_Interface.Rev10_5_2.@`. After these archives have been restored they are of no use and may be deleted.

- `!Machine.Release.X_Interface.Rev10_5_2.@`. If you are never going to install additional terminal types this may be deleted. Do not delete this if you think additional workstations or terminal types will be used.

Installation using SunOS with MIT X11.R3

The steps involved in getting the rxi program running on a sun workstation/file-server running MIT X11R3 are:

❒ 1. Pick an RXI terminal type to install.

If all of the Sun workstations that will be using RXI to connect to R1000's have the older Type 3 keyboard (no numeric pad on the right, no help key on the left) then go to step 2 and install the **XSun3_X11R3** or the **XSun3_X11R3_Sparc** terminal type.

If all of the Sun workstations that will be using RXI to connect to R1000's have the newer Type 4 keyboard (numeric pad on the right instead of the R1-R15 keys, help key on the left) then go to step 2 and install the **XSun4_X11R3** or the **XSun4_X11R3_Sparc** terminal type.

If you will have a mixture of Type 3 and Type 4 keyboards that will all be using the same installed copy of RXI then you will have to do one of two things. Either install RXI as an **XSun3** terminal type or as an **XSun4** terminal type.

If you install RXI as an **XSun3** then:

- Type 3 keyboard users do nothing extra.
- Type 4 keyboard users should put the **rxsun4** command into their personal **~/.rxrc** file (or they can run the command by hand in any shell window prior to the time they log into an R1000). They should also edit their **~/.Xdefaults** file to contain a line that looks like this: **RXI*recognition:xsun4**.

If you install RXI as an **XSun4** then:

- Type 3 keyboard users should edit their **~/.Xdefaults** file to contain a line that looks like this: **RXI*recognition:xsun3**.
- Type 4 keyboard users will have the **rxsun4** command performed for them by the default **rxrc** file in the **app-defaults** directory. If they have a personal **~/.rxrc** file then they will need to place the **rxsun4** command in that file as well.

The line **RXI*recognition:xsun3** or **RXI*recognition:xsun4** tells RXI that when an R1000 asks the question "What type of terminal are you?", it should answer "xsun3" or "xsun4". RXI is built with a default answer that is determined by the **Rxi_Install.XSun?_X11R?** terminal type that you chose to install. This is a way to override that default.

The **rxsun3** and **rxsun4** commands read the **XSun3-xmodmap** and **XSun4-xmodmap** files respectively. These files contain keyboard definitions for the Type 3 and Type 4 keyboards. These commands are intended only for the use of users with Type 4 keyboards.

Note: If the Type 3 keyboards are using one file server and the Type 4 keyboards are using another then you can simply install RXI as an **XSun3** on the one server and as an **XSun4** on the other server. Treat the two servers as two separate RXI installations as

shown above. The R1000's should all have both the **XSun3** and the **XSun4** Editor_Data files installed.

X11 from MIT has a "problem" with Type 4 keyboards. SunOS 3.? does not know about Type 4 keyboards and so it always thinks that users have Type 3. SunOS 4.? knows about Type 4 keyboards and so the X server will realize that a Type 4 keyboard is in use, however, the MIT server for Suns has a default keyboard mapping that makes most of the keys in the rightmost keypad useless; they don't transmit anything when pressed unless the Shift key is also being held.

Since the R1000 enjoys the use of these keys this problem had to be addressed. The **rxsun4** command tells the X server to redefine the entire keyboard. Every key on the keyboard is assigned to be a key of some type. Typically the new type is the same as the old type; the differences lie in the area where the default keyboard mapping has various keys defined as NoSymbol (no-operation). This allows a Type 4 user to access all of his keys.

Please note that this is a "global" effect. It affects not just RXI but rather globally effects every applicaton being run upon the workstation. At some point either MIT or Sun will come out with a proper default keyboard mapping that does not make the extra keys NoSymbol; at that time the **XSun4** terminal type will be changed to reflect the new defaults and the **rxsun4/rxsun3** commands will disappear.

The **rxsun3** command turns a Type 4 keyboard into a Type 3 keyboard. Any "new" keys (keys that are new with the Type 4 keyboard) are marked as NoSymbol (no-operation). This is another option for Type 4 keyboard users; they can turn their keyboard into a Type 3.

Special Installation Note for Type 4 keyboard installations:

The **XSun3-xmodmap** and **XSun4-xmodmap** files should be tested prior to installation by the **make install.all** step. While the X server is running, issue the following two commands:

if on a Type 4 keyboard :

```
xmodmap XSun3-xmodmap
xmodmap XSun4-xmodmap
```

if on a Type 3 keyboard :

```
xmodmap XSun4-xmodmap
xmodmap XSun3-xmodmap
```

If you get any messages of the general form "bad keycode value (out of range)" then you are using a version of the X server that contains a minor bug. The problem is that the server has some tables inside of it for describing the keyboard. The tables are 1 entry short. To fix the problem just edit the **XSun3-xmodmap** and **XSun4-xmodmap** files. Search for the line (there is one in each file) that has "keycode 132" on it. Comment out that line by placing a "!" at the front of the line. This will eliminate the problem and you can proceed with the next installation step. This bug means that your numeric keypad "+"

key will not be available for use; the server does not realize that it exists.

❒ 2. Create a source directory for the rxi program on the workstation/file-server. For example, if the X11 sources from MIT are located in a directory called **/src/x** then the suggested place for rxi would be **/src/x/clients/rxi**. If you would prefer to put the sources into a different place, eg. **/vendor/src/rxi**, then that works as well.

❒ 3. Traverse to **!Machine.Release.X_Interface.Rev10_5_2** on the R1000.

❒ 4. Run the **Rxi_Install.Workstation_Files** procedure; this gets the files copied to the workstation/file-server from the R1000. Use the **Rxi_Install.XSun3_X11R3**, **Rxi_Install.XSun3_X11R3_Sparc**, **Rxi_Install.XSun4_X11R3** or **Rxi_Install.XSun4_X11R3_Sparc** terminal types. For example:

```
Rxi_Install.Workstation_Files
     (Keyboard              => Rxi_Install.Xsun3_X11r3,
      On_Machine            => "nova",
      Username              => "sagan",
      Password              => "verbose",
      Account               => "unlimited",
      Rxi_Source_Directory => "/src/x/clients/rxi");
```

❒ 5. Connect to the source directory on the workstation and do the command **make restore**. This unpacks rxi and makes it available for complete installation.

If you are installing the **XSun4_X11R3** terminal type then you need to edit the file named **rxrc** in the source area.

Toward the beginning of the **rxrc** file, insert a new line so that the first command that is executed by the **rxrc** script is the **rxsun4** command. The file probably starts with a command like:

```
xset m 3 2
```

This sets the mouse acceleration factors. You should add the **rxsun4** command before that line.

```
rxsun4
xset m 3 2
```

❒ 6. If this site has some or all of the optional licensed Rational products then edit the **RXI_Env_Menu** file. It's format is that of a C source file so it should look familiar. There is a #define line for each licensed product. For each licensed product at this site, set the corresponding #define to be equal to 1. This activates the rxi Environment menu entries for that product.

❒ 7. At this point you have two choices. A) Handle rxi in the typical MIT fashion using the "imake" and "makedepends" programs, or, B) handle rxi as an independent stand-alone program.

a. If your site uses the imake/makedepends programs for X11.R3 based applications then do:

if the sources are in **/src/x/clients/rxi** as suggested execute:

```
make -f Makefile.MIT Makefile
```

if the sources are in some other directory then do something like:

```
make -f Makefile.MIT Makefile TOP=/src/x
```

Substitute the appropriate main X11.R3 source directory for **/src/x**. This will regenerate the **Makefile** in the typical MIT fashion. Follow that with a **make depend** command.

b. If your site does not use the image/makedepends programs, or if you do not want to use them in this case then you may wish to hand-edit the Rational-supplied **Makefile**. At the top of the **Makefile** is a list of the source, library, include, and font paths that will be used during installation. If you wish to change any of these default locations please do so now. The defaults are:

BINDIR - where rxi executables will reside;
eg. **/usr/bin/X11**
FONTDIR - where the server fonts reside;
eg. **/usr/lib/X11/fonts**
MANDIR - the man page directory for the rxi man page;
eg **/usr/man/mann**
XAPPLOADDIR - where application defaults files reside;
eg. **/usr/lib/X11/app-defaults**

❒ 8. FOR NCD INSTALLS ONLY: Edit the file **Makefile** and add **/ncd** to the end of the pathname for the **FONTDIR** entry.

❒ 9. You now have three more choices. A) Install the Rational-supplied rxi as-is, or, B) relink to use local (possibly shared) libraries, or C) completely recompile and recreate rxi using all local definitions. The supplied version of rxi should be able to execute on your system. However, it will not use any shared libraries, nor will it incorporate any local library fixes, extensions, or modifications that may exist.

a. If you wish to just use rxi as supplied then skip this step.

b. If you wish to simply relink rxi so that it uses local libraries then do a **make relink** command and proceed to the next step.

c. If you wish to recreate rxi from scratch then do a **make clean** command followed by a **make all**.

❒ 10. Switch users to "super-user" and do a **make install.all**; this will:

- Install the rxi program, the RXI default resource file, and any support programs needed by RXI users. (The command **make install** performs just this step.)

- Install the rxi man page. (The command **make install.man** performs just this step.)

- Compiles the various fonts (**fixed-screen-*.bdf**) and places them into **$(FONTDIR)/bdf/misc** directory so that the X server can find them. These are the normal and bold-face fonts used by rxi. (The command **make install.fonts** performs just this step.)

If you wish to add these fonts to the collection of fonts that came with X11.R3 from MIT (so that they will always be automatically regenerated whenever the MIT fonts are regenerated then issue the command **make install.fonts.source**.

Then connect to the **$(FONTSRC)/bdf/misc** directory and add the appropriate lines to the **Imakefile** there to cause compilation and installation of the new **fixed-screen-*-*.bdf** files. A **bdf.Imakefile** is supplied by Rational which shows the type of entries to be made. Do a **make all** and a **make install** there to compile and install the new fonts on your system.

❒ 11. Install the termcap definition for the rxi terminal type into your local termcap database. There is a file called **termcap** in the rxi source area that contains the termcap definition for rxi. (There is also a **terminfo** file if you are using terminfo on your system.) The **termcap** file is inserted into your local **/etc/termcap** file and the **terminfo** file is used by issuing the command **tic terminfo**.

❒ 12. If the rxi files were installed in the **/src/x/clients/rxi** hierarchy as suggested in step #2 then you will probably wish to edit the **Imakefile/Makefile** in the **/src/x/clients** directory. Add the rxi directory to the **SUBDIRS=...** list. This will cause rxi to be remade and reinstalled whenever all MIT X11 clients are remade.

After installing rxi, at any time:
If an X Server is already running then the **rxi** program may be run directly.
If an X Server is not already running then the **rx** program may be used to run the X Server and then invoke a "console" window, a clock window, and one rxi window on the workstation.

Installation using SunOS with X11/NeWS

The steps involved in getting the rxi program running on a Sun workstation/file-server running XNews are:

❒ 1. Pick an RXI terminal type to install.

If all of the Sun workstations that will be using RXI to connect to R1000's have the older Type 3 keyboard (no numeric pad on the right, no help key on the left) then go to step 2 and install the **`XNews3_V1_0`** or **`XNews3_V1_0_Sparc`** terminal type.

If all of the Sun workstations that will be using RXI to connect to R1000's have the newer Type 4 keyboard (numeric pad on the right instead of the R1-R15 keys, help key on the left) then go to step 2 and install the **`XNews4_V1_0`** or **`XNews4_V1_0_Sparc`** terminal type.

If you will have a mixture of Type 3 and Type 4 keyboards that will all be using the same installed copy of RXI then you will have to do one of two things. Either install RXI as an **`XNews3`** terminal type or as an **`XNews4`** terminal type.

If you install RXI as an **`XNews3`** then:

- Type 3 keyboard users do nothing extra.
- Type 4 keyboard users should edit their **`~/.Xdefaults`** file to contain a line that looks like this: **`RXI*recognition:xnews4`**.

If you install RXI as an **`XNews4`** then:

- Type 3 keyboard users should edit their **`~/.Xdefaults`** file to contain a line that looks like this: **`RXI*recognition:xnews3`**.
- Type 4 keyboard users do nothing extra.

The line **`RXI*recognition:xnews3`** or **`RXI*recognition:xnews4`** tells RXI that when an R1000 asks the question "What type of terminal are you?", it should answer "xnews3" or "xnews4". RXI is built with a default answer that is determined by the **`Rxi_Install.XNews?_V1_0`** terminal type that you chose to install. This is a way to override that default.

Note: If the Type 3 keyboards are using one file server and the Type 4 keyboards are using another then you can simply install RXI as an **`XNews3`** on the one server and as an **`XNews4`** on the other server. Treat the two servers as two separate RXI installations as shown above. The R1000's should all have both the **`XNews3`** and the **`XNews4`** Editor_Data files installed.

NOTE to R1000 users using the model 4 keyboard with X11/NeWS:

Sun has seen fit to make some keys on the keyboard "identical".

In particular, the keys "Stop" and "F11" as well as the keys "Again" and "F12" are defined, by X11/NeWS, to be the same X "key symbol". This means that the R1000 cannot tell the difference between a user typing what he thinks of as "Again" and what he thinks of as "F12".

In addition the Alternate key is defined as NoSymbol which means that it does nothing. In order to use the Alternate key with an R1000 you will need to issue the command **`xmodmap -e 'keycode 26 = Alt_R'`**. This will define the Alternate key and make it available for use. This will have to be done once each time you log into the Sun.

❒ 2. Create a source directory for the rxi program on the workstation/file-server. For example, if your various X11 application sources are located in **`/src/x`** then a likely place would be **`/src/x/rxi`**.

❒ 3. Traverse to **`!Machine.Release.X_Interface.Rev10_5_2`** on the R1000.

❒ 4. Run the **`Rxi_Install.Workstation_Files`** procedure; this gets the files copied to the workstation/file-server from the R1000. Use the **`Rxi_Install.Xnews3_V1_0`**,**`Rxi_Install.Xnews3_V1_0_Sparc`**, **`Rxi_Install.Xnews4_V1_0_Sparc`** or the **`Rxi_Install.Xnews4_V1_0`** terminal types. For example:

```
Rxi_Install.Workstation_Files
     (Keyboard              => Rxi_Install.Xnews3_V1_0,
      On_Machine            => "nova",
      Username              => "sagan",
      Password              => "verbose",
      Account               => "unlimited",
      Rxi_Source_Directory => "/src/x/clients/rxi");
```

❒ 5. Connect to the source directory on the workstation and do the command **`make restore`**. This unpacks rxi and makes it available for complete installation.

❒ 6. If this site has some or all of the optional licensed Rational products then edit the **`RXI_Env_Menu`** file. It's format is that of a C source file so it should look familiar. There is a #define line for each licensed product. For each licensed product at this site, set the corresponding #define to be equal to 1. This activates the rxi Environment menu entries for that product.

❒ 7. Edit the file called **`Makefile`**. There are a series of "make" macros at the beginning of the file. These macros tell the installation script where the various pieces of RXI are to be installed. The default values for these macros may be correct for your site but they should be checked. The macros are:

`OPENWIN` - the "home" directory for Open Windows (X/NeWS);
eg. **`/openwin`**
`BINDIR` - where rxi executables will reside;
eg. **`/openwin/bin`**
`FONTDIR` - where the server fonts reside;
eg. **`/openwin/lib/fonts`**
`MANDIR` - the man page directory for the rxi man page;
eg. **`/usr/man/mann`**
`XAPPLOADDIR` - where application defaults files reside;
eg. **`/openwin/etc`**

❒ 8. FOR NCD INSTALLS ONLY: Edit the file **Makefile** and add **/ncd** to the end of the pathname for the **FONTDIR** entry.

❒ 9. You now have three more choices. A) Install the Rational-supplied rxi as-is, or, B) relink to use local (possibly shared) libraries, or C) completely recompile and recreate rxi using all local definitions. The supplied version of rxi should be able to execute on your system. However, it will not use any shared libraries, nor will it incorporate any local library fixes, extensions, or modifications that may exist.

a. If you wish to just use rxi as supplied then skip this step.

b. If you wish to simply relink rxi so that it uses local libraries then do a **make relink** command and proceed to the next step.

c. If you wish to recreate rxi from scratch then execute the **make clean** command followed by a **make all**.

❒ 10. Switch users to "super-user" and execute **make install.all**; this will:

- Install the rxi program, the RXI default resource file, and any support programs needed by RXI users. (The command **make install** performs just this step.)
- Install the rxi man page. (The command **make install.man** performs just this step.)
- Compile the various fonts (**fixed-screen-*.bdf**) into the **$(FONTDIR)** directory and then does a "bldfamily" there. These are the normal and bold-face fonts used by rxi. (The command **make install.fonts** performs just this step.)

❒ 11. Install the termcap definition for the rxi terminal type into your local termcap database. There is a file called **termcap** in the rxi source area that contains the termcap definition for rxi. (There is also a **terminfo** file if you are using terminfo on your system.) The **termcap** file is inserted into your local **/etc/termcap** file and the **terminfo** file is used by issuing the command **tic terminfo**.

Installation using AIX and IBM X-Windows

The steps involved in getting the rxi program running on an IBM PC/RT workstation/file-server are:

❒ 1. Create a source directory for the rxi program on the workstation/file- server. For example, if program sources are usually kept in a directory called **/src** then within this directory create a directory called **/src/rxi**.

❒ 2. Traverse to **!Machine.Release.X_Interface.Rev10_5_2** on the R1000.

❒ 3. Run the **Rxi_Install.Workstaion_Files** procedure; this gets the files copied to the workstation/file-server from the R1000. For example:

```
Rxi_Install.Workstaion_Files
       (Keyboard              => Rxi_Install.Xrtus_Aix2_1,
        On_Machine            => "My_Machine",
        Username              => "MyName",
        Password              => "MyPassword",
        Account               => "",
        Rxi_Source_Directory => "/src/rxi");
```

❒ 4. Connect to the source directory on the workstation and do the command **make restore**. This unpacks rxi and makes it available for complete installation. For example:

```
cd /src/rxi
make restore
```

❒ 5. If this site has some or all of the optional licensed Rational products then edit the **RXI_Env_Menu** file. It's format is that of a C source file so it should look familiar. There is a #define line for each licensed product. For each licensed product at this site, set the corresponding #define to be equal to 1. This activates the rxi Environment menu entries for that product.

❒ 6. Edit the file called **Makefile**. A particular site may wish to change the places where rxi, its fonts, and/or its man page are installed. These can be changed by changing the definition of:

BINDIR - where the rxi shell script goes; this is the twin to the aixterm script;
usually /usr/bin

X11BINDIR - where the real rxi goes; this is where most X11 executables go;
usually /usr/lpp/X11/bin

XAPPLOADDIR - where the default resource file goes;
usually /usr/lpp/X11/defaults

FONTDIR - where the system-wide X11 fonts live;
usually /usr/lpp/fonts

MANDIR - where the system-wide "new" man pages live;
usually /usr/man/mann

❒ 7. You now have three more choices. A) Install the Rational-supplied rxi as-is, or, B) relink to use local (possibly shared) libraries, or C) completely recompile and recreate rxi using all local definitions. The supplied version of rxi should be able to execute on your

system. However, it will not use any shared libraries, nor will it incorporate any local library fixes, extensions, or modifications that may exist.

a. If you wish to just use rxi as supplied then skip this step.

b. If you wish to simply relink rxi so that it uses local libraries then do a **`make relink`** command and proceed to the next step.

c. If you wish to recreate rxi from scratch then do a **`make clean`** command followed by a **`make all`**.

❒ 8. Turn yourself into "super-user" and do a **`make install.all`**; this will:

- Install the rxi program, the RXI default resource file, and any support programs needed by RXI users. (The command **`make install`** performs just this step.)
- Install the rxi man page. (The command **`make install.man`** performs just this step.)
- Install the rxi terminfo description. This allows terminfo-using programs (such as vi) to operate within an rxi window. (The command **`make install.terminfo`** performs just this step.)
- Compile the various fonts **`fixed-*.bdf into the $(FONTDIR)`** directory. These are the normal and bold-fase fonts used by rxi. (The command **`make install.fonts`** performs just this step.)

Some sites may not want to use the Rational supplied fonts. They can skip the **`install.man`** and **`install.fonts`** steps if they wish. They will want to edit the **`RXI.Xdefaults`** file to change the names of the default rxi fonts.

Some sites may not have the optional licensed IBM "man" product and they can skip the **`install.man`** step.

Installation using Vax VMS and DECwindows

The steps involved in getting the rxi program running on a DEC VAXstation workstation/file-server are:

❒ 1. Create a source directory for the rxi program on the workstation/file- server. For example, **User:[Rxi]**.

❒ 2. Traverse to **!Machine.Release.X_Interface.Rev10_5_2**.

❒ 3. Run the **Rxi_Install.Workstaion_Files** procedure; this gets the files copied to the workstation/file-server from the R1000. Specify the main directory (eg. **User:[Rxi]**) as the destination of the transfer. For example:

```
Rxi_Install.Workstaion_Files
        (Keyboard                =>
Rxi_Install.Xdecus_Vms5_1_Multinet203,
         On_Machine              => "My_Machine",
         Username                => "MyName",
         Password                => "MyPassword",
         Account                 => "",
         Rxi_Source_Directory => "User:[Rxi]");
```

❒ 4. Do a **SET DEFAULT** to the source directory on the workstation and do the command **@makefile restore**. This unpacks rxi and makes it available for complete installation.

❒ 5. If this site has some or all of the optional licensed Rational products then edit the **RXI_Env_Menu** file. It's format is that of a C source file so it should look familiar. There is a #define line for each licensed product. For each licensed product at this site, set the corresponding #define to be equal to 1. This activates the rxi Environment menu entries for that product.

❒ 6. Edit the **MAKEFILE.COM** file in the source area. Follow the editing instructions at the front of the file. You will be asked to specify the TCP/IP library to use (Excelan MultiNet or Wollongong WIN TCP/IP) and to specify the directories containing the library include files and the library .OLB file. This is also the place where you indicate the proper installation directories and protections for the various pieces of the RXI system.

❒ 7. You now have three more choices. A) Install the Rational-supplied rxi as-is, or, B) relink to use local (possibly shared) libraries, or C) completely recompile and recreate rxi using all local definitions. The supplied version of rxi should be able to execute on your system. However, it will not use any shared libraries, nor will it incorporate any local library fixes, extensions, or modifications that may exist.

a. If you wish to just use rxi as supplied then skip this step.

b. If you wish to simply relink rxi so that it uses local libraries then do a **@makefile relink** command and proceed to the next step.

c. If you wish to recreate rxi from scratch then do a **@makefile clean** command followed by a **@makefile all**.

❒ 8. Enable your privileges; **BYPASS** should be sufficient. Do an **@makefile install.all**; this will:

- Installs the rxi program, its helper program rxi_detached, and the RXI default resource file. (The command **@makefile install** performs just this step.)
- Install the rxi help library. (The command **@makefile install.help** performs just this step.)
- Compiles and installs the various fonts **fixed-screen-*.bdf** in the **FONTDIR** directory (defaults to **SYS$COMMON:[SYSFONT.DECW.USER_75DPI]**). These are the normal and bold-face fonts used by rxi. (The command **@makefile install.fonts** performs just this step.)

❒ 9. Edit the **SYS$MANAGER:SYLOGIN.COM** file and add a definition for a system-wide symbol named RXI. eg.

```
$ RXI :== $ SYS$COMMON:[SYSEXE]RXI_DETACHED.EXE
```

This creates a system-wide foreign command that will run rxi as a detached process. You will have to logout and log back in to get it defined for yourself.

❒ 10. Reboot any and all DECwindows workstations (or just the servers) that need to use rxi. The reboot is necessary in order to cause the servers to recognize the new rxi fonts.

❒ 11. You may wish to do a **PURGE** on: **SYS$SYSTEM:**, **DECW$SYSTEM_DEFAULTS:**, and **SYS$COMMON:[SYSFONT...]**.

Feedback

To help us improve our instructions and make installations easier, we provide the following form for your feedback. Please complete and send to:

Product Assembly and Test
Rational
3320 Scott Blvd.
Santa Clara, CA 95054-3197
Atten: Mark Dutra

❒ 1. How long did the installation take you?

❒ 2. Did the installation proceed as described in the instructions?

❒ 3. What did you like about this set of instructions/installation?

❒ 4. What did you dislike about this set of instructions/installation?

❒ 5. What could be done to improve these (or other) instructions/installations?

Environment

Installation Procedure

D_12_8_0

RATIONAL

PN 503-003207-16A

This document is subject to change without notice.

Rational Software Corporation, 2800 San Tomas Expressway, Santa Clara, California 95051-0951

1 Installation

Overview

This package is composed of:

- ❒ 1. D_12_8_0 Environment Tools tape
- ❒ 2. D_12_8_0 DFS tape
- ❒ 3. These installation instructions
- ❒ 4. D_12_8_0 Environment Release Information

When installing the Environment release, you can view the upgrade process as a series of phases. These phases must be performed in a serial manner.

- ❒ 1. Assessing the impact of the release on the user community
- ❒ 2. Verifying prerequisites to the installation
- ❒ 3. Preparing the user community and the machine
- ❒ 4. Loading the release
- ❒ 5. Cleaning up

These phases must be performed in a serial manner, as must the steps in each of the phases. This Installation Procedure is organized by phase, with each phase comprising a subsection of this document. Once you have performed this installation, please fill out and return the feedback form that is the last page of these instructions along with a printout of the file `Do_Step_Execution_Time` found in the `Logs` library of the release archive. Your feedback is very important in helping us to improve our installation instructions.

A number of steps are automated through the use of a standard procedure called `Do_Step`. If you are unfamiliar with the use of this procedure, refer to Appendix A for further information and the format of the instructions below.

NOTE: A number of the steps will display a filtered error log. When these are displayed, a line containing an ellipsis (. . .) is appended. If this is the only line you see in the display, it indicates there were no errors.

The following sections take you through the installation process one phase at a time. Be sure to complete ALL of the phases of installation and to perform these phases in the order in which they appear here.

Warning: *Follow these installation steps in a serial manner. Do not attempt any other work or even to look at objects on the system (such as Environment specs) while performing this installation. All users must be logged off during the installation. Failure to adhere to these instructions may result in the failure of portions of the installation.*

Upgrade Time Estimates

The time required for the installation of this release will vary, depending on which Environment version the machine is currently running:

❒ 1. Upgrading from releases prior to D_12_7_3

If the machine is currently running a release earlier than D_12_7_3, the upgrade will require that the D_12_7_3 upgrade first be applied, and then the D_12_8_0 upgrade.

❒ 2. Upgrading D_12_7_3 or D_12_7_4

If the machine is currently running D_12_7_3 (Delta 3.1) or D_12_7_4 (Delta 3.1a), the upgrade will require at least 6 hours of system time during which no users can have access to the machine.

In planning for the upgrade, the following rough time frame can be anticipated:

❒ 1. The first steps of the upgrade up to and including **RELEASE_RESTORE** will take approximately an hour for D_12_7_3 or D_12_7_4 upgrades.

❒ 2. Next, the **Install_Product** step will take 1 to 4 hours to complete (1 hour for a "virgin" Environment with no user data, 4 hours for a machine heavily loaded with user data).

During the last part of the **Install_Product** step, the new DFS will be automatically installed from the Environment level.

During the **Install_Product** step, the machine can be unattended, and progress messages will be displayed which you can check periodically. This step can be left to run overnight, if desired.

❒ 3. Next, the machine is shutdown, and if the new DFS was not loaded automatically during the `Install_Product` step, the new DFS is loaded from tape. This takes about half an hour using an 8mm tape drive, or about an hour using a 9-track tape drive.

❒ 4. Next, the EEPROMs are updated and the machine is rebooted. This reboot takes about half an hour longer than usual because the new subsystems are installed. This step can be left to run overnight, if desired.

❒ 5. Finally, some checking and cleanup steps are performed, which take about an hour.

At the time this step is run, the upgrade is virtually complete, and users may be allowed on the system, and work while this step completes.

NOTE: If you are upgrading multiple machines that are connected via a network a tool exists for upgrading all other machines over the network, after the first machine is upgraded. These subsequent upgrades over the network will take significantly less time and require little operator intervention. Refer to Appendix B for details.

Assessing the Impact of the Release

In this phase of the installation, you will assess the impact of the release on the user community and the machine.

❒ 1. When installing a new Environment release or a new layered product release, private activity files may become obsolesced (i.e., the subsystems they reference may not be compatible with the upgraded software). When installing several products at once, the Environment release should be installed first. Subsequently, each layered product can be installed. Following this process will result in the system default activity (`!Machine.Release.Current.Activity`) referencing a compatible set of subsystems.

After the upgrades have been completed, each user should merge the contents of the default activity into his or her own private activity file with the command `Activity.Merge`.

❒ 2. Several Environment Ada objects will be overwritten with newer versions. The old versions of modified bodies are maintained in a file of the name `<objectname>_Vn` where `n` is the version number of the original object. After the upgrade is completed, these saved files can be used to merge local customizations back into the new version, and then the file can be deleted.

❒ 3. The machine will be rebooted during this installation.

- ❒ 4. The D_12_8_0 release includes an archive for the Rational Access product. After the upgrade, the nested archives can be found in the `!Machine.Release.Archive` world. Rational Access may be installed using the Rational Access installation procedure, shipped as part of the D_12_8_0 release. Contact Order Administration for additional copies. (See the associated product Release Information to determine if these products need to be installed; note that if these products are to be installed, the `LOAD_TAPE` step of the product installation can be skipped as the release archives have already been loaded as part of the D_12_8_0 `LOAD_TAPE` step).

- ❒ 5. Expect this upgrade to use about 100 megabytes of available disk space. Most of this usage will be reclaimed after the archives are deleted and garbage collection has run (see the "Cleaning up" section). If the system does not have this much room, it is possible to selectively restore only the Environment archive (and leave the Rational Access archive on the tape). This reduces the requirement to about 36 megabytes for the Environment archive and an additional 20 megabytes or so to perform the actual installation. See the LOAD_TAPE step for details.

Verifying Prerequisites to the Installation

In this phase of the installation, you will verify that the prerequisites for installing this upgrade have been met. Do not proceed with the upgrade until all prerequisites have been fulfilled.

- ❒ 1. You have read the Release Information.

- ❒ 2. You have read this entire Installation Procedure.

- ❒ 3. The system is executing Environment D_12_7_3 (D3.1) or D_12_7_4 (D3.1a). If the system is executing a release earlier than D_12_7_3 (D3.1), it **MUST** be upgraded to D_12_7_3 (D3.1) before proceeding.

- ❒ 4. You have verified that there is at least 100 megabytes of free *usable*[1] disk space on one of the system disks to account for worst case volume loading. The D_12_8_0 and Rational Access archives will consume ~80 megabytes. Most of this space will be reclaimed after D_12_8_0 and Rational Access have been installed and cleaned up, and garbage collection has run.

Preparing the User Community and the Machine

In this phase of the installation you will prepare both the user community and the machine for the installation.

- ❒ 1. Have a primary backup made.

[1]This space should be measured as that which will not cause garbage collection to start.

❒ 2. Have a full backup made (after the primary).

❒ 3. Verify the full backup tape set using the `Verify_Backup` procedure.

❒ 4. Locate a DFS backup tape. If not available, arrange for one to be made at some time prior to this upgrade.

Warning: *It is extremely important to have these backups available in the event that there is a failure during the upgrade to D_12_8_0. Once started, it is not possible to revert back to the older Environment except by restoring a backup. Taking both primary and full backups gives you two backup recovery options should you need to revert to the previous Environment release.*

Loading the Release

In this phase, you will load the tools required to install the user visible part of the release onto the machine. The release world located in `!Machine.Release.Archive.Environment.D_12_8_0` contains nested archives. This phase of the installation takes several hours.

❒ 1. Load the Environment D_12_8_0 tools tape onto the tape drive.

❒ 2. Log in on an account which is a member of group *Privileged.*

The following steps utilize the procedure `Do_Step` as described in Appendix A.

❒ 3. The Environment D_12_8_0 tape contains a release archive for the Rational Access product; this archives will use about 40 megabytes of additional disk if is is restored as is the case with the **LOAD_TAPE** step (for a total of just under 80 megabytes being restored by the **LOAD_TAPE** step). If disk space is of no concern, skip the remainder of this step and execute the **LOAD_TAPE** step that follows. To selectively restore just the Environment archive (which requires about 40 megabytes for the archive data) do not execute the **LOAD_TAPE** step; rather, create a command window and execute the following command:

```
if Operator.Privileged_Mode then
    Archive.Restore (Objects =>
                        "!Machine.Release.Archive.Environment.D_12_8_0?");
else
    Log.Put_Line ("Cannot enable privileged mode");
end if;
```

After the Environment D_12_8_0 has been successfully installed and cleaned up, you may execute either of the following to restore the associated product archive for use with its installation procedure included as part of this release. Note that you should skip the **LOAD_TAPE** step of the product installations.

```
if Operator.Privileged_Mode then
--  Restore RATIONAL_ACCESS
    Archive.Restore (Objects =>
                "!Machine.Release.Archive.RATIONAL_ACCESS.Release1_0_3?");
else
    Log.Put_Line ("Cannot enable privileged mode");
```

```
end if;
```

Skip the **LOAD_TAPE** step and continue on with the step which follows it.

❒ 4. **LOAD_TAPE** *[30 Minutes]*

This loads the contents of the tools tape. Unlike the other steps in this installation procedure, this step can be executed from any command window (as opposed to being executed from **!Machine.Release.Archive.Environment.D_12_8_0** which does not yet exist). Answer the mount request at the operator console. There should be no errors.

NOTE: If the machine does not have the **Do_Step** procedure, use **Archive.Restore** to load the tape.

NOTE: This step also loads the Rational Access archive, as described in the "Assessing the Impact of the Release" section above. This is loaded in the **!Machine.Release.Archive** world.

❒ 5. Go to **!Machine.Release.Archive.Environment.D_12_8_0**.

❒ 6. **AUTHORIZATION_CHECK** *[1 Minute]*

This step will verify that the appropriate products are authorized for this machine. These products are :

```
CMVC
CMVC.Source_Control
```

If there are products which need to be authorized, this will be indicated and a command window created with a call to procedure **Product_Authorization.Register**. Modify the appropriate parameters and authorize the products required.

❒ 7. **USERS_CHECK** *[1 Minute]*

This step sets the login limit to 1 and verifies that there are no other users logged into the system. Note that the login limit will be restored to the original value when the machine is rebooted later during the installation.

❒ 8. **PREPARE_MACHINE** *[5 Minutes]*

This step will delete all EEDB configurations except the current (elaborated) configuration,

and then delete all subsystems not referenced by the current configuration. It will then reclaim (destroy) all code segments that no longer belong to a subsystem (note: the number of code segments reclaimed will be displayed on the system console in a message of the form: **`EEDB: Deleted: xxxx`**).

No action will be taken by this step if any member of the running configuration has **`XXX`** as its version. If this is detected, a warning message about an illegal version of a "below the line" subsystem will be generated, asking you to continue with the installation (if no other errors are detected by this step). Refer to Appendix C for the procedure to fix the problem and reclaim the unused code segments.

❒ 9. **RELEASE_RESTORE**

[15 Minutes]
This step restores the **`Release`** archive. When completed, a filtered error log is displayed (**`Restore_Release_Log_Summary`** located in **`!Machine.Release.Archive.Environment.D_12_8_0.Logs`**). Examine this log for errors. Ignore the following types of errors (if present):

```
!!! can't restore link...

!!! Creating CMVC database ...
    no history is currently
    being saved.

++* Can't resolve default
    switch file ...

++* can't set subclass for ... to
    TEXT (ILLEGAL_OPERATION).
```

❒ 10. **INSTALL_PRODUCT**

[1 ~ 4 Hours]
This step executes the **`Install_Product`** procedure located in **`!Machine.Release.Environment.D_12_8_0`**. The machine can be left unattended while this step is running.

The following messages will be displayed in a window so that progress of the step may be monitored:

```
92/10/02 15:02:11 +++ Installing Environment Release D_12_8_0.

---- Beginning Process: Updating the Segments in the Elaborator Database
```

```
------- Beginning step: Loading New Code Segments
...

------- Beginning step: Updating the Design_Facility Subsystem
...

------- Beginning step: Updating the Mail Subsystem
...

------- Beginning step: Checking the Configuration in the Elaborator Database
...
```

NOTE: If you encounter an error in one of the above steps where a **<name>.DELTA** subsystem cannot be found, refer to Appendix C for a procedure to recover. In most cases, the **INSTALL_PRODUCT** step will continue to completion and the problem can then be corrected.

```
---- Beginning Process: Updating the Universe Specs

------- Beginning step: Restoring New Universe Items
...

------- Beginning step: Restoring a Prototype Universe (takes about an hour)
...

------- Beginning step: Inserting New Declarations Incrementally
...

------- Beginning step: Changing Default Values of Selected Command Parameters
...

------- Beginning step: Scanning for Changes in the Universe Specs
...
15:51:53 >>> !COMMANDS.WORK_ORDER'SPEC is NOT a forced unit, therefore it
15:51:53 ... will be added to Units_To_Be_Updated.
...
15:52:55 >>> !IMPLEMENTATION.DIRECTORY'SPEC is NOT a forced unit, therefore
15:52:55 ... it will be added to Units_To_Be_Updated.
15:52:58 >>> !IMPLEMENTATION.DIRECTORY has 34 IMMEDIATE dependents ( 33
15:52:58 ... specs):
15:52:58 >>>     !TOOLS.SYSTEM_UTILITIES'SPEC'V(8).
15:52:58 >>>     !IO.DEVICE_INDEPENDENT_IO'SPEC'V(1).
15:52:58 >>>     !IO.IO'SPEC'V(1).
15:52:58 >>>     !TOOLS.PROFILE'SPEC'V(2).
15:52:58 >>>     !COMMANDS.LOG'SPEC'V(2).
15:52:58 >>>     etc...
...
15:54:11 >>> !MACHINE.EDITOR_DATA.ENABLE_PRODUCT_KEYMAPS'SPEC is NOT a
15:54:11 ... forced unit, therefore it will be added to Units_To_Be_Updated.
...


---- Beginning Process: Installing Changed Universe Units

------- Beginning step: Computing Compilation Dependencies for Updated_Units
...

------- Beginning step: Saving ACLs on Updated_Units
...

------- Beginning step: Demoting Clients of Updated_Units
...

------- Beginning step: Replacing Updated_Units
```

```
...

------- Beginning step: Installing Demoted Clients of Updated_Units
...

------- Beginning step: Coding Demoted Clients of Updated_Units
...

------- Beginning step: Restoring Acls on Updated_Units
...

------- Beginning step: Running Snapshot Daemon


---- Beginning Process: Updating Miscellaneous Universe Items

------- Beginning step: Restoring Miscellaneous System Components
...


------- Beginning step: Authorizing Lrm_Interface


------- Beginning step: Authorizing X Interface


------- Beginning step: Enabling 'Install_Subsystems' to run on next reboot.


------- Beginning step: Running Snapshot Daemon


---- Beginning step: Updating Dfs Files

------- Beginning step: Installing Dfs
...
```

You will see the following message after the new DFS files are loaded:

```
16:23:28 +++ The new Dfs has been installed.  You may now reboot the machine
16:23:28 ... using Reboot_For_New_Release.
```

When completed, a window of **!Machine.Release.Environment.D_12_8_0.Logs** is displayed.

There are 2 logs of interest, **Install_Product** and **Install_Universe**, each which has a time stamp as part of the name and a filtered version ending in **_Errors** and **_Negatives**.

❒ Examine **Install_Product_@_Errors** for errors. There should be none.

❒ Examine **Install_Universe_@_Errors** for errors. There should be none. See the next step if there are errors.

NOTE: It is important to note whether the new DFS was automatically loaded in the **Install_Product** message shown above. If the DFS was not installed, you will have to load it from tape in a later step.

❒ 11. Updating Universe Specs

If there are no errors in these files, skip to the next step. If there are errors, it may be that not all attempts to update the Universe specs were completed automatically. With a few exceptions, if a Universe spec has any dependents, it will **NOT** be changed by Install_Product (the exceptions are named in the file `!Machine.Release.Environment.D_12_8_0.Lists.Forced_Units`).

In some cases it is possible to manually insert a specification which failed to be inserted automatically. Reference the `Install_Product` log for the text to be inserted in these cases. Only address problems identified by `***` or `++*` entries in the summary error logs. After the upgrade has been completed, you will address other cosmetic problems identified by `>>>` in the original log. Do **NOT** at this point attempt to make all specs consistent with the D_12_8_0 version. All spec changes in this release are compatible with the existing specs in the sense that this release will run with the old specs.

`Install_Product` may be rerun as many times as you like. Each run will generate a new log that indicates variances with the "official" versions of the Universe.

❒ 12. Before continuing, you must know whether the new DFS was installed automatically as part of the `Install_Product` step (see the description of the final message in the `Install_Product` step for the message).

If you are unsure whether the DFS was automatically installed or wish to verify, examine the log file `!Machine.Release.Environment.D_12_8_0.Logs.Install_Product_<date>_At_<time>` and check the end of the file for these messages:

```
...
12:25:20 --- File LOCAL_STANDARD.M200_CONFIG being written.
12:25:21 --- File LOCAL_PATCH.M200_CONFIG being written.
12:25:21 --- File LOCAL_EEDB.M200_CONFIG being written.
12:25:21 --- File LOCAL_KERNEL.M200_CONFIG being written.
12:25:21 --- File DDC.M200_CONFIG being written.
```

If these messages are present, the DFS was automatically installed. If these message are not present, the DFS must be loaded from tape in the following steps.

❒ 13. Turn the `OPERATOR MODE` keyswitch to the `INTERACTIVE` position.

❒ 14. On Series 400 machines only, toggle the `EPROM WRT PRT` switch on the RESHA board to the off (down) position. (You have to pull out slightly on the switch in order to flip it down.) When this switch is in the off position, the `PRT OFF` light on the RESHA board will be on.

❒ 15. Go to `!Machine.Release.Archive.Environment.D_12_8_0`.

❒ 16. **SHUTDOWN**

[10 Minutes]
This step executes `Reboot_For_New_Release` located in `!Machine.Release.Environment.D_12_8_0.Load_Procs`. `Reboot_For_New_Release` verifies that all keymaps are installed, verifies that all initialization procedures are coded, turns off `Archive_On_Shutdown`, sets the shutdown warning delay to 5 minutes, and then initiates the shutdown procedure.

If there are any keymap files which are not in the installed state, the shutdown will not take place. If there are errors promoting keymaps, determine if the given keymaps are required to be in the installed state. If not, shutdown the machine manually using the `Schedule_Shutdown` procedure.

After this step has completed and the machine has been successfully scheduled to shutdown, you may log off; note that your session will be automatically logged off when the machine shuts down if you do not log off prior to shutdown.

❒ 17. If the new DFS was automatically installed during the `Install_Product` step (see above), boot the machine and skip to step 19.

If the new DFS was not installed during the `Install_Product` step, continue with the next step to load it from tape.

❒ 18. Mount the D_12_8_0 DFS tape onto the system tape drive.

Loading DFS *[30 minutes]*

You will boot the system from the DFS tape, which will automatically run the `RECOVERY` program.

Note that:

- `UPDATE_EEPROM` no longer needs to be run manually as a separate step after loading the DFS for series 200/300/400: EEPROMs are updated automatically as part of the `RECOVERY` program.
- `CEDIT` no longer needs to be run after loading the DFS: the hardware configuration file is saved.

Sample console dialogue is shown below with required responses shown in bold type.

The sample dialogue shown (starting with the **Shutdown** step) is for a Series 400S machine. Filenames and messages may vary slightly for different processors and configrations. Messages that may differ have been marked with # in the sample below.

```
  =====>> Environment Log <<=====
  13:25:15 +++ Operator Archive_On_Shutdown FALSE by OPERATOR.S_1
  13:25:15 +++ Operator Shutdown_Warning  5:00.000 by OPERATOR.S_1

  =====>> Daemons (System Job 5) <<=====
  from System: 1:25:21 PM; System will shutdown in  5:00.000
  from System: 1:29:21 PM; System will shutdown in  1:00.000

  =====>> Environment Log <<=====
  13:30:21 +++ Shutdown_Task Disabling_Terminals

  =====>> Daemons (System Job 5) <<=====
  13:30:33 +++ Shutdown_Task Killing_Job  4: System
  13:30:33 +++ Shutdown_Task Killing_Job  5: Daemons
  13:30:33 +++ Shutdown_Task Killing_Job  6:
  13:30:33 +++ Shutdown_Task Killing_Job  7:
# 13:30:34 +++ Shutdown_Task Killing_Job  240: Archive Server
# 13:30:34 +++ Shutdown_Task Killing_Job  244: Print Spooler
# 13:30:35 +++ Shutdown_Task Killing_Job  247: Smooth Snapshots
# 13:30:35 +++ Shutdown_Task Killing_Job  248: Console Command Interpreter

# =====>> Console Command Interpreter (System Job 248) <<=====
  13:30:45 +++ Shutdown_Task Starting_Om_Shutdown

# =====>> Console Command Interpreter (System Job 248) <<=====
  13:30:51 +++ Low_Level_Action.Abandon_For_Shutdown No_outstanding_actions

  =====>> CONFIGURATOR <<=====
  starting virtual memory shutdown
  starting snapshot

# =====>> Console Command Interpreter (System Job 248) <<=====
  snapshot is finished
# virtual memory shutdown at ( 57, 28-FEB-92 13:30:55)
  system shutdown is complete

  Sequencer has detected a machine check.
  ***********************************************
  R1000 IOP Boot requested from PC=0001AD94, Boot_Reason code = 0C
  R1000 Halt or Machine Check detected

# Restarting R1000-400S February 28th, 1992 at 14:35:44
  Change BOOT/CRASH/MAINTENANCE options [N] ? y
  Enable MODEM dialout [Y] ? <RETURN>
  Enable MODEM answer [Y] ? <RETURN>
  Enable I/O Processor AUTO BOOT [Y] ? n
  Enable R1000 AUTO BOOT [N] ? <RETURN>
  Enable AUTO CRASH RECOVERY [N] ? <RETURN>
  Enable CONSOLE BREAK KEY [Y] ? <RETURN>
  Are your answers correct [Y] ? <RETURN>
  Are these new defaults [N] ? <RETURN>
  Change IOP ENVIRONMENT configuration [N] ? <RETURN>

# Logical tape drive 0 is an 8mm cartridge tape drive.
# Logical tape drive 1 is declared non-existent.
```

```
# Logical tape drive 2 is declared non-existent.
# Logical tape drive 3 is declared non-existent.
  Booting I/O Processor
# IOC Series 400 Bootstrap Version 0.4

  Boot from (Tn or Dn) [D0] t0
  <If using tape drive 3, enter t3>

# Tape_Boot_4_0_7 July 23, 1991 at 7:24:34 AM
# Waiting for tape unit ready.
# Strike any key to abort.....................rewinding

  Select files to boot [D=DEFAULT, O=OPERATOR_SUPPLIED] : [D] <RETURN>
  Skipping..
# Loading FS_0.M200

# Loading RECOVERY.M200
# Skipping..............................
# Loading M400S_KERNEL_0.M200

# Initializing M400S I/O Processor Kernel 4_2_12
  Disk  0 is ONLINE and WRITE ENABLED
# Disk  1 is ONLINE and WRITE ENABLED
# Disk  2 is ONLINE and WRITE ENABLED
  <number of disk drives may be different>
  IOP Kernel is initialized
  Enable line printer for console output [N] ? <RETURN>
      RECOVERY 14.01 92/05/03 10:00:00\
  Options are:
      0 => Exit
      1 => Initialize disk (for experts only)
      2 => Initialize disk, drop USR defects (internal use only)
      3 => Show MFG and USR bad block locations
      4 => Show only USR bad block locations
      5 => Install new DFS only
      6 => Show bad block count and DOS limits
  Enter option : 5
  Enter unit number of disk to format/build/scan (usually 0) : 0
# HDA : J1181
  Writing bad block information.
  Writing boot label.
  Writing DFS label.
  Constructing free list.
  Writing free list.
  Allocating and initializing directory.
  Creating predefined files.
  KERNEL_0.M200
  KERNEL_1.M200
  KERNEL_2.M200
  FS_0.M200
  FS_1.M200
  FS_2.M200
  PROGRAM_0.M200
  PROGRAM_1.M200
  PROGRAM_2.M200
  DFS_BOOTSTRAP.M200
  ERROR_LOG
  Tape drive unit number : 0
  <If using tape drive 3, enter 3>
# Reading -> DFS_BOOTSTRAP.M200
# Reading -> KERNEL_0.M200
# Reading -> FS_0.M200
# Reading -> RECOVERY.M200

  < ... 3200+ files are loaded ... >
```

```
# Reading -> LOCAL_STANDARD.M200_CONFIG
# Reading -> LOCAL_PATCH.M200_CONFIG
# Reading -> LOCAL_EEDB.M200_CONFIG
# Reading -> LOCAL_KERNEL.M200_CONFIG
# Reading -> DDC.M200_CONFIG
# Elapsed time is 00:10:00
  The DFS files have been reloaded
  Restored TCP_ID_HOST_ID file containing 89.64.64.56
```

NOTE: If an address of 255.255.255.255 is restored, but this is not the correct IP address of the machine, do not be alarmed. This DFS-level TCP_IP_HOST_ID file is not used for network communications. As long as the Environment-level file **!Machine.Tcp_Ip_Host_Id** has the correct IP address for the machine, all is well.

❒ 19. Updating the EEPROMs *[5 minutes]*

At this point, the D_12_8_0 DFS is loaded. If you just read the DFS from tape, the **RECOVERY** program will automatically run the **UPDATE_EEPROM** program.

If the DFS was loaded during the **INSTALL_PRODUCT** step and you skipped the previous step, rebooting the machine will automatically run the **UPDATE_EEPROM** program.

In all cases, the **UPDATE_EEPROM** program automatically updates the out-of-date EEPROMs and reboots the machine again. Below is sample console dialogue for a Series 400S machine. Filenames and messages may vary slightly for different processors and configrations. Messages that may differ have been marked with # in the sample below.

```
#     UPDATE_EEPROM 5.4 92/10/16 12:34:56\
# Comparing the new prom files to the actual EEPROMs:
# SELFTEST: prom revision 910812, file revision 921105 - PROM needs updating
# BOOT    : prom revision 910812, file revision 921105 - PROM needs updating
# UTIL/DEB: prom revision 910808, file revision 921105 - PROM needs updating
# NET/DEB2: prom revision 901204, file revision 921105 - PROM needs updating
  not updating IOC board revision #
  {The following 4 EEPROMs are not found on Series 200 or 300 machines.}
# RES_TEST: prom revision 901218, file revision 920603 - PROM needs updating
# LANCE   : prom revision 901218, file revision 920603 - PROM needs updating
# DISK    : prom revision 901218, file revision 920603 - PROM needs updating
# TAPE    : prom revision 910808, file revision 920603 - PROM needs updating
  not updating RESHA board revision #
# updating the EEPROMS will take about 243 seconds to complete.
# turn EEPROM WRITE PROTECT switch (at front of RESHA board) OFF (down) - OK
  {On Series 200 and 300 machines, the message above will not appear.}
# Updating EEPROM SELFTEST
# EEPROM successfully updated
# Updating EEPROM BOOT
# EEPROM successfully updated
# Updating EEPROM UTIL/DEB
# EEPROM successfully updated
# Updating EEPROM NET/DEB2
# EEPROM successfully updated
  {The following 4 EEPROMs are not found on Series 200 or 300 machines.}
# Updating EEPROM LANCE
```

```
# EEPROM successfully updated
# Updating EEPROM DISK
# EEPROM successfully updated
# Updating EEPROM TAPE
# EEPROM successfully updated
  {On some machines, you may see messages like the following:
       # Warning from REE_UPLOAD: prom write timed out - will retry
       # Warning: 5251 bytes were incorrectly written
       # Warning from REE_UPLOAD: prom write timed out - will retry
       # Warning: 4883 bytes were incorrectly written
       # Warning from REE_UPLOAD: prom write timed out - will retry
       # Warning: 4289 bytes were incorrectly written
       # Warning from REE_UPLOAD: prom write timed out - will retry
       # Warning: 2901 bytes were incorrectly written
       # Warning from REE_UPLOAD: prom write timed out - will retry
       # Warning: 1406 bytes were incorrectly written
       # EEPROM successfully updated and verified, ignore the warnings above
  As long as the "successfully updated" message appears, there is no problem.}
# Updating EEPROM RES_TEST
# EEPROM successfully updated
# turn the EEPROM WRITE PROTECT switch ON (up) - OK
  {On Series 200 and 300 machines, the message above will not appear.}
# all EEPROMs are up to date now

  ************************************************
  Booting R1000 IOP after updating the EEPROMs
    Boot Reason code = 20, from PC 0001ADA2

# Restarting R1000-400S November 28th, 1992 at 15:02:27

  OPERATOR MODE MENU - options are:
      1 => Change BOOT/CRASH/MAINTENANCE options
      2 => Change IOP CONFIGURATION
      3 => Enable manual crash debugging (EXPERTS ONLY)
      4 => Boot IOP, prompting for tape or disk
      5 => Boot SYSTEM

  Enter option [Boot SYSTEM] : <RETURN>

# Logical tape drive 0 is an 8mm cartridge tape drive.
# Logical tape drive 1 is declared non-existent.
# Logical tape drive 2 is declared non-existent.
# Logical tape drive 3 is declared non-existent.
  <Your tape configuration may be different.>
  Booting I/O Processor with Bootstrap Version 0.4

# Initializing M400S I/O Processor Kernel 4_2_16
# Disk  0 is ONLINE and WRITE ENABLED
# Disk  1 is ONLINE and WRITE ENABLED
# Disk  2 is ONLINE and WRITE ENABLED
# Disk  3 is ONLINE and WRITE ENABLED
  <number of disk drives may be different>
  IOP Kernel is initialized
  Initializing diagnostic file system ... [OK]
  ============================================================
  Restarting system after automatically updating EEPROMs

# >>> NOTICE: the EEPROM WRT PROT switch is OFF (at front of RESHA board) <<<
  {On Series 200 and 300 machines, the message above will not appear.}
  >>> NOTICE: the OPERATOR_MODE switch is set to INTERACTIVE <<<

  --- Booting the R1000 Environment ---
```

NOTE: This reboot will take about half an hour longer than usual because new subsystems are installed. This step can be left to run overnight, if desired.

During this reboot, you will see the following messages regarding predefined users, which can be ignored. They only appear during the first reboot after installing D_12_8_0 and should not appear again:

```
19:01:29 !!! Predefined_Users User_create_failed : Public: User already exists
19:01:30 !!! Predefined_Users Prohibit_login_failed Delete status for
             Public.S_1 : NAME_ERROR
19:01:30 !!! Predefined_Users User_create_failed : Network_Public: User
             already exists
19:01:30 !!! Predefined_Users Prohibit_login_failed Delete status for
             Network_Public.S_1 : NAME_ERROR
19:01:30 !!! Predefined_Users User_create_failed : Operator: User already exists
19:01:30 --- Predefined_Users Created_group : Privileged
19:01:30 --- Predefined_Users Created_group : Mailer
19:01:31 --- Predefined_Users Created_group : Spooler
19:01:31 !!! Predefined_Users User_create_failed : Rational: User already exists
19:01:31 --- Boot Running "!Machine.Initialization".Start
19:01:31 --- Predefined_Users Created_group : System
19:01:31 --- Boot Running Destroy deleted Ada/Link objects
19:01:33 +++ Predefined_Users Created_Session !Machine.Network_Public_Session
19:01:33 +++ Predefined_Users Created_Session !Machine.Public_Session
19:02:55 +++ Predefined_Users Created_World !Machine.
             Network_Public_Archive_Server_Sessions
19:02:55 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_1
19:02:56 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_1
19:02:56 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_2
19:02:58 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_2
19:02:58 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_3
19:02:59 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_3
19:02:59 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_4
19:03:00 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_4
19:03:00 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_5
19:03:01 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_5
19:03:01 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_6
19:03:02 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_6
19:03:03 +++ Predefined_Users Attempting_to_create_session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_7
19:03:03 +++ Predefined_Users Created_Session !Machine.
             Network_Public_Archive_Server_Sessions.Network_Public_Session_7
```

❒ 20. Turn the **OPERATOR MODE** keyswitch to the **AUTOMATIC** position.

❒ 21. On Series 400 machines only, toggle the **EPROM WRT PRT** switch on the RESHA board to the **WRT PRT** (up) position. (You have to pull out slightly on the switch in order to flip it up.) The **PRT OFF** light will not be on if this switch is in the correct position.

❒ 22. Log in on an account which is a member of group *Privileged.*

❒ 23. Verifying Install_Subsystems

Go to the library **!Machine.Release.Environment.D_12_8_0.Logs** and examine the **Install_Subsystems_@_Errors**. Ignore the following types of errors, if present:

```
*** Tool state for ... has been lost.
+++ Severe problems encountered while processing view ...
    These have been repaired...
```

In some cases a subsystem state library may be frozen, prohibiting restoration of the compatibility database and new specs. This condition may be identified by an error message of the following form:

```
*** Policy error opening compatibility database unit map for ...
%%%  promotion of ... has terminated abnormally with
     exception !Lrm.System.Nonexistent_Page_Error, ...
```

If this occurs, unfreeze the state library and its contents, then restore the subsystem manually from the subsystem archive library located in **!Machine.Release.Environment.D_12_8_0.Archives.Subsystems**. For example:

```
Operator.Enable_Privileges;
if Operator.Privileged_Mode then
    Archive.Restore (Objects => "!Commands.System_Maintenance",
                     Options => "Replace,Promote",
                     Device => "Archives.Subsystems");
else
    Io.Put_line ("User does not have privileges.");
end if;
```

You then need to check/promote units in the spec/load view which may have been left in an uncoded state from the failed restore attempt.

❒ 24. Verifying Environment Specs

Go to **!Machine.Release.Environment.D_12_8_0.Logs** and examine the **Install_Product_<date>_<time>** log, searching for log prefixes of the type ">>>" which annotate units that should be updated but cannot due to dependencies.

With a few exceptions, if an Enviroment spec has any dependents, it will **NOT** be changed by **Install_Product** (the exceptions are named in the file **!Machine.Release.Environment.D_12_8_0.Lists.Forced_Units**).

The log entries prefixed with ">>>" show the differences between what is on your machine and what is in the new release. The differences are in the style of **File_Utilities.Difference**, where file 1 is the spec installed on your machine and file 2 is the spec for this release.

In most cases, the differences are cosmetic or minor (comments or parameter lists reformatted, pragmas moved the beginning to the end of a file, etc.). Such changes can be ignored, as they will not affect Environment commands, and are not worth the large amount of time (up to 18 hours or more) required to recompile them and their dependent units.

The entries prefixed by ">>>" should be checked to verify that they all represent cosmetic or minor changes.

Up to the first 5 IMMEDIATE dependents are identified in the log. If the last line of this list is "etc...", there are more than five immediate dependents. The total number of specs and bodies that are immediate dependents is indicated in the log as well.

You will probably find at least two units that could not be updated (unless they were recompiled when a previous Environment release was installed):

```
19:14:49 >>> !COMMANDS.WORK_ORDER'SPEC is NOT a forced unit, therefore it
19:14:49 ... will be added to Units_To_Be_Updated.
19:14:55 >>> !COMMANDS.WORK_ORDER'SPEC has x IMMEDIATE dependents ( y
19:14:55 ... specs):
. . .

19:16:07 >>> !IMPLEMENTATION.DIRECTORY'SPEC is NOT a forced unit, therefore
19:16:07 ... it will be added to Units_To_Be_Updated.
19:16:27 >>> !IMPLEMENTATION.DIRECTORY'SPEC has x IMMEDIATE dependents ( y
19:16:27 ... specs):
. . .
```

If these are the only units which could not be updated, you may skip to the next step. If there are other units which could not be updated, you and the system manager will have to decide whether to demote and recompile them.

You may also find entries prefixed with ">>>" which do not list any dependents and do not list any differences. For example:

```
01:11:22 >>> !MACHINE.EDITOR_DATA.ENABLE_PRODUCT_KEYMAPS'SPEC is NOT a
01:11:22 ... forced unit, therefore it will be added to
01:11:22 ... Units_To_Be_Updated.
```

These messages can be ignored.

If you find specs with significant differences or specs with minor changes that you want to install, a procedure has been provided in the release to do this (even if the specs have clients):

```
"!Machine.Release.Environment.D_12_8_0.Load_Procs".Install_Universe
            (Units_List  => "$$.Lists.Units_To_Be_Updated",
             Device      => "$$.Archives.Universe");
```

Execute this command in the context of the **!Machine.Release.Environment.D_12_8_0** directory.

The `Units_List` parameter is the name of a file containing a list of units to be installed. The default for this parameter designates a file that was constructed by the `Install_Product` procedure.

❒ 25. Go to `!Machine.Release.Archive.Environment.D_12_8_0`.

❒ 26. **RESTORE_NOTES** *[1 Minute]*
This step restores the D_12_2_4, D_12_5_0, D_12_6_5, and D_12_8_0 Release Information to `!Machine.Release.Release_Notes`, and the `Guide_To_Machine_Initialization` to `!Machine.Initialization`. The Guide is simply a stand-alone version of Appendix E, "Machine Initialization," in the *System Manager's Guide* delivered with this release, and is installed here in both PostScript and line printer forms, as a convenience to the user.

❒ 27. **RECORD_INSTALLATION** This step records the release information in `!Machine.Release.Current.Products`. This is a required step and establishes that the release has successfully been installed.

❒ 28. Any customizable procedure (i.e., which had a body) which was replaced by this release will have the original procedure saved to a text file, located in the same library, under the name `<procedure>_[Body, Spec]_Vn`, where n is the version. For example, the unit `!Commands.Abbreviations.Print'Body` would be saved as `Print_Body_V4`. Work with the system manager to merge any modifications saved in these files into the new procedures. For a list of units which were overwritten see the file `Lists.Forced_Units` in the release library `!Machine.Release.Environment.D_12_8_0`.

❒ 29. Go to `!Machine.Release.Archive.Environment.D_12_8_0.Logs` and print out the file `Do_Step_Execution_Time`. Return this printout to SMSE along with other feedback supplied on the last page of this procedure.

Cleaning up

In this phase, you will perform some cleanup activities.

This should not be done if this machine will be used to distribute the release to other machines on the network. See Appendix B.

❒ 1. **DESTROY_ARCHIVE** This step destroys the release archive library in `!Machine.Release.Archive`. Only execute this step after the D_12_8_0 release has been running successfully for several days and you are confident there are no problems. There will be

an error when an attempt is made to set the context back to the release archive library. This can be ignored.

- ❒ 2. Destroy older release libraries located in `!Machine.Release.Environment`. It is recommended that the latest release library (`D_12_8_0`) be maintained for possible future use, especially if there were any errors during the installation, but this is not a requirement and this library may also be deleted.

- ❒ 3. Have customer take a full Environment backup.

Post-Installation Customer Checklist

- ❒ 1. Make a full backup of the Environment.

- ❒ 2. Make a backup of the DFS if the D_12_8_0 DFS tape was not left at the site (secure sites only).

- ❒ 3. Each user should merge the contents of the default activity into his or her own private activity file with the command **Activity.Merge**.

```
Activity.Merge (Source => "!Machine.Release.Current.Activity",
                Mode => Activity.Differential,
                Target => ">>User Activity<<");
```

  Note that this operation may have undesirable side affects of changing subsystem references which were explicitly set by the user.

- ❒ 4. Distribute the Release Information to the user community. A PostScript and line printer copy of the D_12_8_0 Release Information can be found in **!Machine.Release.Release_Notes**.

- ❒ 5. Update the **!Machine.Editor_Data.Daily_Message** file indicating that Environment D_12_8_0 has been installed. Include information about previous items in this checklist when appropriate. If applicable to the user community, announce the availability of the LRM Interfaces.

- ❒ 6. Set the ACLs on new specs/files restored as part of this release, if desired.

- ❒ 7. If desired, arrange for installation of the Rational Access product. Installation archives were installed in **!Machine.Release.Archive** as part of the D_12_8_0 upgrade, and installation procedures included with the D_12_8_0 release.

A Procedure Do_Step

The majority of the steps which must be performed for this installation utilize the procedure `Do_Step` (included as part of the release tape and located in `!Commands`) to execute the required sequence of commands which implement the step. This procedure **must always** be executed from a command window in the release library

```
!Machine.Release.Archive.Environment.D_12_8_0
```

Each step which utilizes procedure `Do_Step` will be of the form

<STEP_NAME> *[<TIME TO EXECUTE>]*
<DESCRIPTION>

where `<STEP_NAME>` is passed as the parameter to procedure `Do_Step` which performs the necessary actions to complete the step, `<TIME TO EXECUTE>` is the amount of time the step takes to execute (this may be expressed as a range, in which case the lower values is typically the minimum, and the upper value is an estimate of the maximum), and `<DESCRIPTION>` is a description of what the step does, including any action which you will need to manually perform. For example:

FOO *[5 Minutes]*
This step is an example of the format used for steps which are executed using procedure `Do_Step`. To execute step **FOO**, you would go to `!Machine.Release.Archive.Environment.D_12_8_0` and in a command window, execute:

```
Do_Step ("FOO");
```

which would result in automatic execution of all commands required to implement this step. If you needed to take any manual action in addition to executing `Do_Step`, it would be noted as :

❒ Perform some manual check

If any errors occur for this step, you should always fix the problem before proceeding on to the next step. Each step is defined by a fragment of Ada code which is executed by `Program.Run`. These fragments are stored in the `Steps` file located in the `Command_Data` library of the

release. In the event you want to modify a step, you can use the "PROMPT => <STEP>" form when invoking `Do_Step`. See the spec of procedure `Do_Step` (located in `!Commands`) for more detailed information.

Warning: *If you interrupt the execution of* `Do_Step` *(by using* `Job.Interrupt` *such as* `CONTROL-C`*) it is possible that certain interactive commands executed by some steps, such as* `Common.Definition`*, may fail with an exception and display a message such as*

```
Unable to read file due to Constraint_Error (Null Access)
```

In general, this message has no negative impact on the execution or completion of the step, and can be ignored.

Note: *When multiple steps are executed, you will be prompted between steps with the following:*

```
Continue with <Step Name> step (Continue | Skip | Quit)?  [Continue] : [input]
```

Select the action to be taken, one of `Continue`*,* `Skip`*, or* `Quit`*.* `Continue` *(the default if* `PROMOTE` *is entered without typing anything) results in the step <Step Name> being executed.* `Skip` *will skip <Step Name> and prompt with the step following <Step Name>.* `Quit` *results in termination of* `Do_Step` *without further steps being executed.*

B Upgrading Over the Network

If you are upgrading multiple machines that are connected via a network AND you are upgrading from D_12_7_3 or D_12_7_4, a tool exists for upgrading all other machines over the network, after the first machine is upgraded. These subsequent upgrades over the network will take significantly less time and require little operator intervention.

Some prerequisites and restrictions on upgrading over the network:

- The machines being upgraded MUST be running D_12_7_3 or D_12_7_4 currently. If you are trying to upgrade from a previous Environment release, you must first install the D_12_7_3 release.

- One of the machines on the network must have been upgraded to D_12_8_0 using the standard method described in the "Loading the Release" section of this document. This machine becomes the 'source' machine for distributing the D_12_8_0 release to the other machines on the network.

- The operator password for the 'source' machine must be known.

- All users must be logged off the machine to be upgraded over the network. The operator should use **`Op.Limit_Login(1)`** to restrict users from logging on during the upgrade. Logins will be reenabled when the machine is rebooted.

When these prerequisites are met, follow these steps on each machine on the network to be upgraded:

❒ 1. **`Archive.Copy`** the **`Network_Install`** procedures from the 'source' machine:

```
Archive.Copy ("!!<'source' machine name>!Machine.Release.Environment." &
              "D_12_8_0.Load_Procs.Network_Install")
```

❒ 2. Execute **`Network_Install`** from the **`!Machine.Release.Environment.D_12_8_0.Load_Procs`** directory:

```
Network_Install (Machine  => "<'source' machine name>",
                 Release  => "D_12_8_0",
                 Username => "Operator",
                 Password => "<Operator password on 'source' machine>");
```

This takes from 1 to 4 hours to run depending on the amount of user data on the machine.

While `Network_Install` runs, progess messages will be displayed. See the messages listed in step number 10 of the "Loading the Release" section of this document.

These messages indicate what step is currently executing and any errors that occur in this execution. This output is also stored in the master log file located in `!Machine.Release.Environment.D_12_8_0.Master_Log_File`. This file will contain the messages from all the steps in the installation.

The complete logs for each step are located in `!Machine.Release.Environment.D_12_8_0.Logs` in the files `Install_Product_<Date and Time>` and `Install_Universe_<Date and Time>`.

These logs should be consulted if errors are found, to determine where in the process these errors occured. See step number 11 of the "Loading the Release" section of this document on how to handle such errors.

The file `!Machine.Release.Environment.D_12_8_0.Lists.Units_To_Be_Updated` contains a list of universe package specs that differ from the "standard" distribution universe. These units will also be reported while `Network_Install` runs, under the step Scanning for Changes in the Universe Specs. These messages are prefixed with '>>>'. See step number 24 of the "Loading the Release" section of this document for information on these messages.

The DFS is installed as part of the `Network_Install` step. If for some reason it is not installed by `Network_Install` (an exception is raised, the machine crashes, etc.), you should install the DFS using the standard tape method. See step numbers 18-21 of the "Loading the Release" section of this document on how to do this.

❒ 3. On Series 400 machines only, toggle the `EPROM WRT PRT` switch on the RESHA board to the off (down) position. (You have to pull out slightly on the switch in order to flip it down.) When this switch is in the off position, the `PRT OFF` light on the RESHA board will be on.

❒ 4. Execute the procedure `!Machine.Release.Environment.D_12_8_0.Load_Procs.Reboot_For_New_Release`.

`Reboot_For_New_Release` verifies that all keymaps are installed, verifies that all machine initialization procedures are coded, turns off Archive_On_Shutdown, sets the shutdown warning delay to 5 minutes, and then initiates the shutdown procedure.

- ❒ 5. Continue with step numbers 19-23 of the "Loading the Release" section of this document (updating the EEPROMs, booting the machine, and verifying **Install_Subsystems**).

- ❒ 6. Use **Archive.Copy** to copy the D_12_8_0 release notes from the 'source' machine:

```
Archive.Copy ("!!<'source' machine name>!Machine.Release.Release_Notes." &
              "Environment_Release12_7_3@");
```

This completes the network upgrade. When all machines at the site have been upgraded, clean up the D_12_8_0 archive on the 'source' machine by running the **Destroy_Archive** step, as described in the "Cleaning up" section of this document.

C Troubleshooting EEDB Configuration Problems

There are two common problems with the machine's EEDB configuration that may cause errors or warnings and which can be handled without home office intervention. These are described below. If you encounter other problems, please contact the Response Center.

❒ 1. If, during the PREPARE_MACHINE step, an illegal version of a "below the line" subsystem is found, the step aborts, preventing the unused code segments from being reclaimed (and thus leaving a small amount of garbage on the disks).

If no other errors are reported by the PREPARE_MACHINE step, you can safely continue with the rest of the installation and at some convenient time in the future when the machine is down, do the following procedure to correct the problem and reclaim the unused segments:

a. Load a tape on the tape drive.

b. Write all **.MLOAD** files to tape:

```
CLI> dump *.mload    -- if using an Exabyte as tape
                        unit 3, use dump/unit=3 *.mload
```

c. Boot the machine:

```
CLI> boot
```

d. When the machine is booted, remount the tape.

e. Read in the **.MLOAD** files:

```
EEDB: read           -- for tape unit 3, use read/unit=3
```

By re-reading in the **.MLOAD** files at the EEDB level, the version number of the subsystems can be determined (e.g., OS_UTILITIES.11.3.0 instead of OS_UTILITIES.XXX).

After doing these steps, you can redo the **Do_Step("Prepare_Machine")** again in order to get rid of the unused subsystems and code segments.

❒ 2. If, during the INSTALL_PRODUCT step, a .DELTA subsystem is missing, the missing subsystem will be reported and you will see several lines of error messages about invalid input (as the rest of the names of the subsystems in the D_12_8_0 configuration are input), and then the step will continue processing.

If there are no other errors reported by INSTALL_PRODUCT, you may rebuild the configuration correctly from the EEDB using the following steps:

- Get to the EEDB, either from the system console or by running **Op.Internal_System_Diagnosis** from an Environment command window. (When running EEDB commands from an Environment window, remember to use the **PROMOTE** key, rather than **RETURN**, to enter commands.)

- Verify how far the new configuration got built (using the "VD $" command).

- DISPLAY <*name*>.+ to get latest version of the missing .DELTA subsystem.

- Use ADD_SUBSYSTEM to add that subsystem version and then the rest of the subsystems "up" the list (see the list of subsystems in the D_12_8_0 release below).

- Continue with the installation after the **INSTALL_PRODUCT** step.

For example, if **DESIGN_FACILITY.DELTA** is missing:

```
EEDB: vd $           -- (make sure it's D_12_8_0 and top line is CMVC)
EEDB: display design_facility.+  -- (get latest number,
                                      e.g., DESIGN_FACILITY.12.0.8D)
EEDB: add_subsystem
Existing Configuration: $
Subsystem.Version: design_facility.12.0.8d  -- (as from output above)
Subsystem.Version: archive.11.3.8d  -- (as from the list below)
Subsystem.Version: native_debugger.11.1.8d
Subsystem.Version: cross_development.delta
Subsystem.Version: initialize.11.2.4d
Subsystem.Version: <return>      -- (ends add_subsystem command)
EEDB: vd $           -- (verify that the subsystems were entered correctly)
```

One can also run
!Machine.Release.Environment.D_12_8_0.Load_Procs.Check_Configuration
to verify that the configuration is now correct. You can now continue with the rest of the installation (assuming **Install_Product** finished--if not, rerun **Install_Product**).

The list of subsystems in D_12_8_0:

```
    INITIALIZE.11.2.4D
    CROSS_DEVELOPMENT.DELTA
    NATIVE_DEBUGGER.11.1.8D
    ARCHIVE.11.3.8D
    DESIGN_FACILITY.DELTA
    CMVC.11.7.2D
    TOOLS_INTEGRATION.DELTA
    FTP_INTERFACE.11.1.2D
    COMMANDS.11.5.3D
    OS_COMMANDS.11.6.1D
    MAIL.DELTA
    OBJECT_EDITOR.11.5.2D
    OE_MECHANISMS.11.1.2D
    TOOLS.11.5.1D
    CORE_EDITOR.11.6.1D
    IMAGE.11.4.2D
    R1000_CODE_GEN.11.50.6D
    R1000_CHECKING.11.50.3D
    R1000_DEPENDENT.11.50.1D
    SEMANTICS.11.50.2D
    COMPILER_UTILITIES.11.50.2D
    INPUT_OUTPUT.11.6.4D
    DIRECTORY.11.4.5D
    PRETTY_PRINTER.11.50.2D
    PARSER.11.50.1D
    DISK_CLEANER.11.1.3D
    ADA_MANAGEMENT.11.50.4D
    BASIC_MANAGERS.11.2.6D
    OM_MECHANISMS.11.1.6D
    NETWORK.11.1.3D
-------------------------------------------------------------------------
    ELABORATOR_DATABASE.11.1.3D
    OS_UTILITIES.11.3.0D
    MISCELLANEOUS.11.1.5D
    ABSTRACT_TYPES.11.2.2D
    ENVIRONMENT_DEBUGGER.10.0.0R
    KERNEL.11.5.7K
    KERNEL_DEBUGGER.11.0.0S
    KERNEL_DEBUGGER_IO.11.0.3
    MACHINE_INTERFACE.11.0.0
    ADA_BASE.11.0.0
    MICROCODE.207.36

where:
    CROSS_DEVELOPMENT.DELTA = CROSS_DEVELOPMENT.11.0.0D
    DESIGN_FACILITY.DELTA = DESIGN_FACILITY.11.4.3D
    MAIL.DELTA = MAIL.11.5.8D
```

D Feedback

In order to help us improve our instructions and make installations easier to do, the following form is provided to allow you to give us feedback. Please complete and send along with a printout of the file `Do_Step_Execution_Time` located in `!Machine.Release.Archive.Environment.D_12_8_0.Logs` to:

Rational
Atten: SMSE
3320 Scott Blvd.
Santa Clara, CA 95054-3197
Re: Environment

❒ 1. How long did the installation take you?

❒ 2. Did the installation proceed as described in the instructions?

❒ 3. What could be done to improve these (or other) instructions/installations?

❒ 4. What did you like about this set of instructions/installation?

Table Of Contents

Rational Environment Training

System Management

Student Notes

RATIONAL

Product Number: 8000-00104

Rev. 2.0, October 1992 (Software Release D_12_6_5)

This document is subject to change without notice.

Rational, 3320 Scott Boulevard, Santa Clara, CA 95054-3197

RATIONAL

System Management Training

1 System Management: Introduction December 1992 RATIONAL

Course Modules

- **Introduction**
- System Overview
- System Operations
- System Administration

2 System Management: Introduction December 1992 RATIONAL

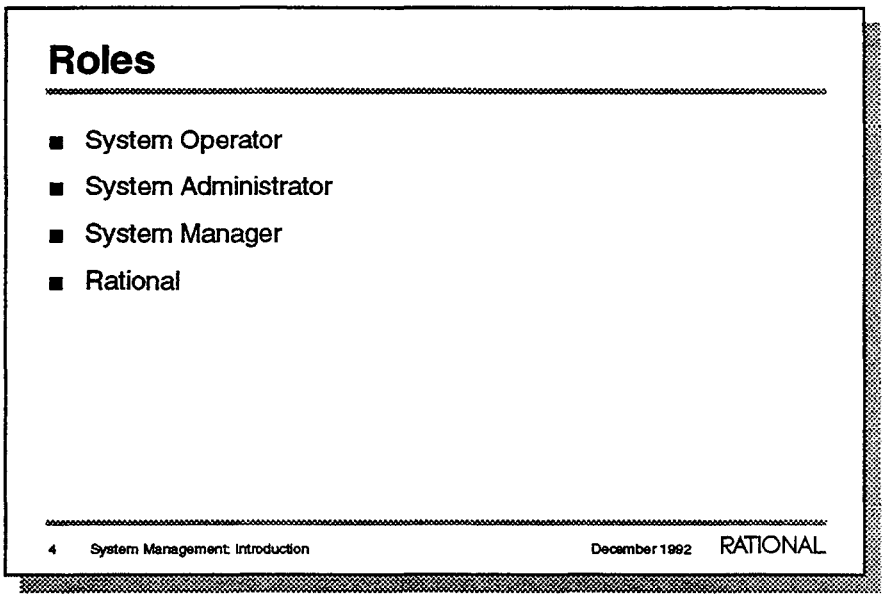
Course Goals

- Describe system operations
- Inform and enable system-management personnel
- Contribute to program success
  - Ensure optimal, efficient use of the R1000
  - Involve system managers in user support

In this course we define R1000 system management and supply you with the background information you need to operate your Rational system effectively and efficiently.

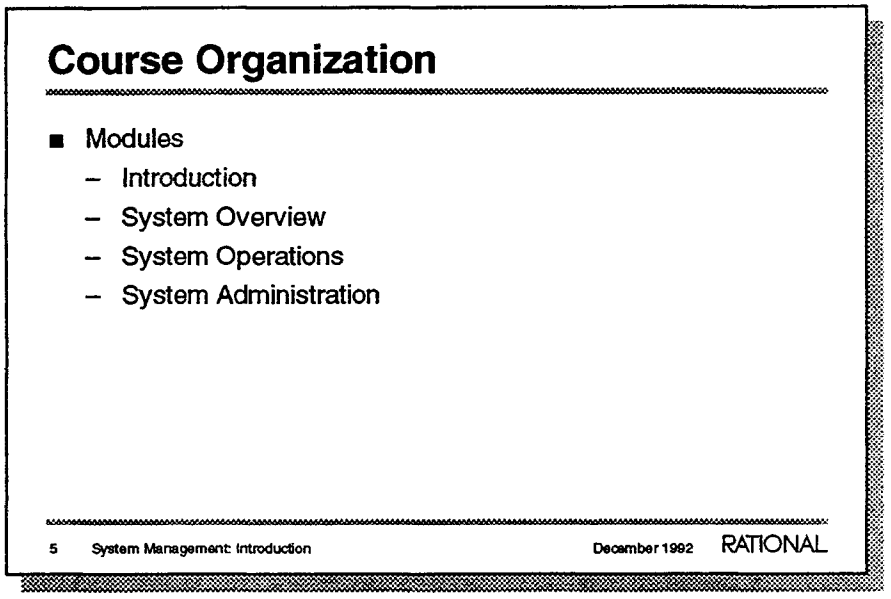
This course is intended to be taught shortly after installation of your machine.

- We'll cover all operations in an overview, and system-management operations in detail.
- Presumed job responsibilities will be presented next. We'll supply conceptual and procedural information.
- This is why the R1000 and the Rational Environment are here, after all!
  - Effective use of system hardware and software resources is important to your company. We want to make it possible for you to employ tools, policies, and processes that complement your organization's working style.
  - As local, knowledgeable screeners of questions and problems, system managers can work closely with Rational support teams to help all users make the best use of the resource.



The course design assumes the job roles listed on this slide. It is entirely possible that all three customer roles are filled by one individual, or that some roles are split among many people at a particular site. As we discuss responsibilities, consider and discuss how these roles are filled at this site.

- The operator is responsible for tasks such as console-based monitoring of system operations, data backup and tape operations, system boot, and system shutdown.
- The administrator is responsible for routine Environment-based tasks such as creating user accounts, monitoring use of system resources, and carrying out access-control and password policies.
- The system manager is responsible for all system wide policies and operations, such as local software tools, helping to set usage policies, and supervising operators and administrators.
- The company is responsible for delivering quality products and an implementation plan to enable project success, and supporting customer activities with expert advice, timely support, and problem-resolution skills.



Here we indicate how the course modules address the topics and audiences described.

- The course is organized into modules so students can identify and if desired attend only appropriate parts, if desired.

  - This module identifies course goals, appropriate audience, and helpful resources, such as documentation. All students should attend this module.

  - The overview module presents a "top-down" view of the system hardware and software, to set the context for following modules. All students should attend this module, even if they will specialize in management of only hardware or software.

  - The operations module supplies basic information about system hardware and low-level software, including configuration options, system boot and shutdown procedures, backup operations, and basic maintenance. This module is for system operators and their supervisors.

  - The administration module indicates how to enable, control and monitor user access of the Rational Environment. It can help you get the most out of your system in a heterogeneous computer network, and prepare you for investigating and reporting problems. This module is for system administrators and system managers.

Course Organization (cont.)

- Structure
  - Information-passing
  - Practice
  - Discussion

- The course consists of lecture-format sessions in which the instructor will transfer information, and lab sessions, in which students will experience and practice their job activities. As time and circumstances permit, there will also be opportunities to discuss and explore options for configuring the system to best meet site requirements.
  - This is accomplished by the instructor talking and/or pointing to written materials and special-purpose handouts.
  - This includes pencil-and paper exercises as well as hands-on access of the system.
  - Student participation is essential if you are to get the most out of the course.

System-Management Duties

- Focal point for
  - R1000 user community
  - Operations personnel
  - Other system/network managers
  - Rational

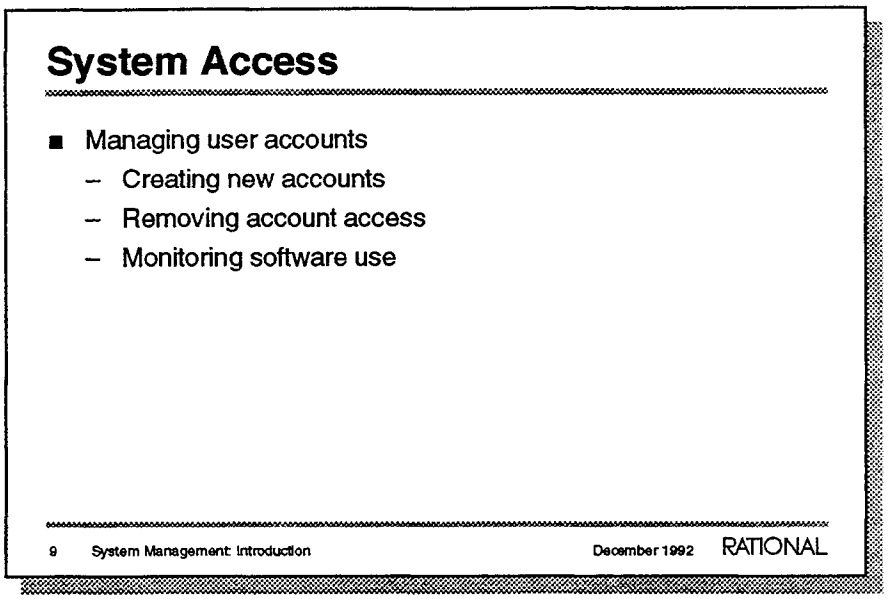
This section explains the duties/tasks performed by the system manager. Later sections describe how to accomplish these. This is one of the key points of the course; you should expect to exit this course with a clear understanding of the responsibilities and duties of a system manager. Typically, more is expected of you, as a Rational system manager, than might be expected of system managers for other computer systems.

- Local system-management personnel can fulfil important coordination roles:
  - The user community typically expects the system manager to coordinate system usage (backups, disk space, system-house keeping functions, and so on) and monitor system performance. At some sites, system managers participate with project leads and/or toolsmiths in defining and implementing local policies related to access controls, standard library structures, use of work orders, site-specific tools, passwords, and so on.
  - If in this class, but not on the system-management team operations personnel must coordinate the backup schedule with the system manager and be trained to notice and communicate system problems.
  - In a typical heterogeneous computing environment, users or system administrators on other machines may be affected by R1000 operations. Examples of these situations include use of shared printer resources, interchange of data via tape, and so on.
  - System and user problems that cannot be resolved locally can be brought to the attention of local Rational Technical Representative or Rational support personnel in the Rational Response Center. We then work with you, the system manager, to achieve closure.

System-Management Duties (cont.)

- Specific duties include:
  - Managing system access
  - Managing system resources
  - Maintaining system availability
  - Coordinating operations activities
  - Interfacing with Rational
  - Communicating with system users

- Each of these is discussed in more detail on the following slides. This is not a comprehensive list of duties—these are typical examples of what Rational system managers do. System managers may:
  - Establish and monitor passwords, user accounts, visibility to data and tools.
  - Monitor and control use of disk space, CPU cycles, and so on.
  - Set and enforce policies to prevent runaway jobs, full disks, and other conditions that may render the system unusable or ineffective.
  - Plan and supervise system backups, inter machine data transfers, preventive maintenance activities, and the like.
  - Represent your management and user community as you track and participate in problem-solving and system-optimization activities.
  - Disseminate information to the user community about system facilities, software availability, and other topics to broad interest.

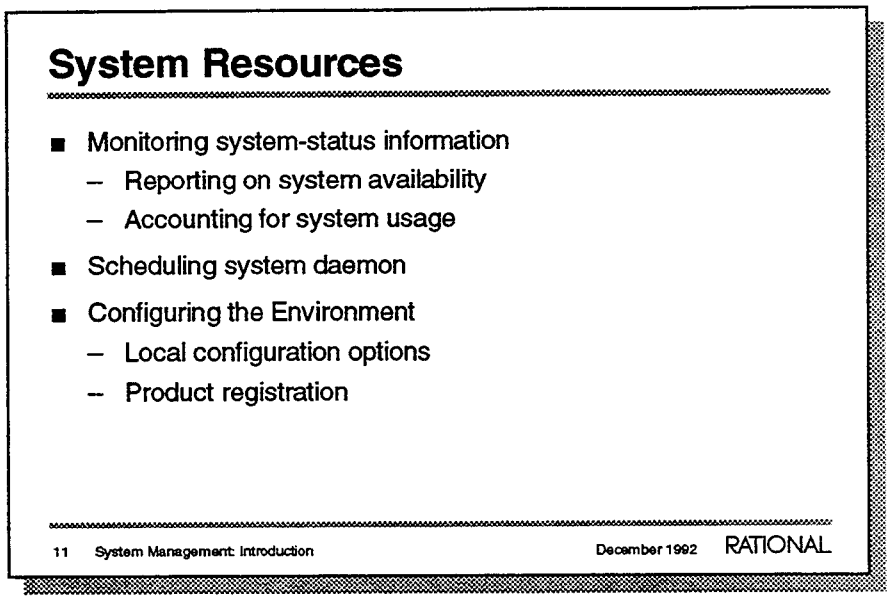


- You must be a privileged user to create accounts for users (logins).
  - Using system wide default values and a single command, the system manager can create a home library and pertinent support structures to enable a new user to log in to the Environment.
  - Maintaining an efficient or secure system can mean revoking a user's right to log in and/or access objects and programs in the Environment.
  - With the Environment and other Rational software products, the number of simultaneous uses (sessions) for the product may be limited. A system manager typically works with company and project management to coordinate use of these resources.

System Access (cont.)

- Controlling access to system information
  - Defining groups
  - Controlling project data
  - Allocating capabilities

- Access to objects and programs by individual users can be controlled
  - For convenience, users may be identified with access-control groups based on working relationship; those groups may then have specific access-control privileges. Typically, the system manager, or another privilege user, maintains the group lists and assigns privileges.
  - In many organizations controlling project data is the responsibility of a project manager, but it other requires coordination with the system manager.
  - When logged in as Operator, special system commands are available. Operator capability can also be extended to other users. Another set of capabilities are available only to members of the group privileged. Typically, the system manager allocates these capabilities by controlling group membership.



All machines have limited amounts of disk and CPU resources are major responsibilities of the system manager.

System managers are expected to collect information about system usage and help implement strategies for optimizing project use of the R1000 and the Environment.

- This involves running built-in programs to log and access a wide range of system-status information.
  - System availability reports include information on system usage, daily disk-space usage, device errors, and general system availability.
  - The machine automatically records system usage by individuals in the user community. Often sites use this information for periodic reports
- The daemon runs client jobs that perform system "housekeeping" functions. Typically, resource-consuming client jobs are scheduled to run automatically during times when user jobs are not likely to be affected
- Many of system characteristics software can be modified to suit local requirements; some modifications are effective only at system-boot time. System-software configuration includes the management of all additional products present at the site, and the coordination of machine resources on a network of several machines, both other R1000s and foreign host.
  - Typically local configuration tasks include updating networking name and address, establishing print queues, and so on.
  - During the boot process, jobs associated with the Rational design facility mail carrier, cross-compiler, or other optional products may need to be registered as system jobs.

System Availability

- I/O Interfaces
  - Terminal ports
  - Network configuration
  - External modems
- Hardware configuration

System managers can monitor and control the availability of R1000 resources. The most basic affects connectivity and configuration of the system itself. In a typical situation, Rational technical representatives help design and install the system interfaces, but post installation responsibility resides with the system manager. Similarly, system manager may perform minor reconfiguration and maintenance operations. System managers who are familiar with the hardware and major interfaces can participate effectively in diagnosing and debugging user/system problems.

- System-level I/O interfaces include those that permit user access to the system and system access to network resources.
  - R1000 system include configurable serial and Telnet ports. For example, a Telnet port supporting a connection with a printer may be configured to disallow logins.
  - In addition to monitoring physical connections, system managers help update R1000 system files and tools with data about network resources, including printers, terminal servers, and computer systems.
  - Depending upon the local network configuration and Rational support contract, your system may use a modem for remote dial-out or dial-in support from Rational. The system manager is the local contact for ensuring access to the resource.
- Your R1000 system may include one or two tape drives, from zero to four disks, and 32 or 64/MB of memory. You should be familiar with the components of your particular system, its maintenance needs, and options for upgrading/reconfiguring the system.

Operations Activities

- Tape-mount request
- Preserving data
  - Backup operations
  - Archive operations
- Stopping and restarting the system

Some R1000 operations may be performed dedicated operations personnel, but they should be overseen by a system manager. R1000 system managers may define or implement local policies for protecting data and controlling system availability.

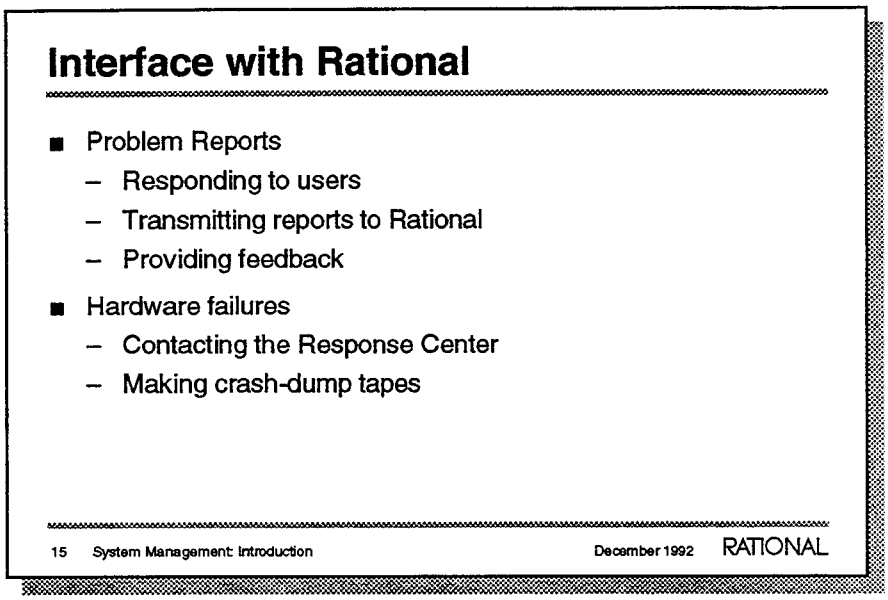
- Magnetic tape is a standard media? for backup and data transmission on R1000 systems. User-initiated requests for data transfer via tape result in tape-mount requests on the system console. System operators respond to the request and mount/dismount tapes.
- Devising and implementing system-level backup policies involves use of two facilities backup and archive.
  - System backup make copies of all the data on a system. Restoring from backup means recreating the data for an entire machine, it is not possible to restore selective data from a backup. Most sites schedule regular system backups; for example, a full backup may be made every week, with incremental backups daily.
  - Archive operations apply to a designated set of objects. These usually represent project state. Data from archive tapes can be restored selectively. Archive operations may be performed by individual user groups, but require the cooperation and coordination of system managers and/or system operators.
- System-management and operations personnel need to know how to safely stop and restart the R1000 system. Such operations may be necessary due to power interruptions or unforeseen emergency situations. Location of power switches, circuit breakers, and system-reset buttons varies with the R1000 model. These are covered in detail later in this course and in your *System Manager's Guide*.

Communicating with Users

- Topics
  - Scheduled backup
  - Planned system downtime
  - New tools/commands
- Techniques
  - Message of the day
  - System broadcast
  - Mail

System managers may need to communicate information about system availability, release upgrades, use of site-specific tools, and so forth. An informed user community is more cooperative with system-management and operations personnel.

- Here we present some typical topics/reasons for communicating with the R1000 users community.
  - Because system performance can degrade noticeably during backup activities, users appreciate knowing when backups are scheduled.
  - Downtime may be anticipated for example, for Rational software release upgrades, preventive maintenance, or building electrical work.
  - Users and system managers may produce programs or command scripts to satisfy a local need. It may be the system manager who arranges and announces their availability to the user community
- There are multiple facilities in the Rational Environment for communicating with users.
  - You can use the what.message command to access a text file for routine messages. Display of the file contents can be made automatic during user-login procedures
  - You can use the Message.Send command interactively to display messages on user screens.
  - You can use the optional Rational Mail product and group aliases to dispatch messages to some or all users who receive electronic mail on the system Such messages can also be sent to users on other systems on the network.



As system manager, you play a key role as focal point for contracting Rational about possible system problems. Typically R1000 users are encouraged to report difficulties to the local system manager, who works with local Rational representatives and/or telephone-based Rational support personnel to solve problems.

- R1000 system software includes an online problem-reporting facility called Request can be used by user.
  - Some reported problems are really misconceptions about the Rational system and Environment software work; solutions may lie in deciding how best to use the patterns of user actions that cause problems, or recommend alternatives to a problematic procedure. Often, some user problems can be addressed directly by the system manager.
  - Working with local Rational technical representatives, system managers can collect, submit, and track reported problems through the Rational Response Center.
  - A system manager gets feedback from Rational about reported problems. Whether they are diagnosed as bugs, user errors, or enhancement requests, you can transmit status information back to your user community. Users work with the system most effectively when they know that potential problems receive the attention they deserve.
- R1000 system hardware typically "belongs" to the system manager.
  - Some problems with the Rational Environment or R1000 system may required contacting the Rational Response Center. Many system managers find this kind of telephone support quite helpful in identifying the real cause of the problem. The Response Center may ask the system manager to perform some basic diagnostic operations to help isolate a problem, or if security allows, the Response may ask to connect to the R1000 through a modem
  - You may be asked to make a crash-dump tape, which captures information about a system crash for later analysis by Rational Support personnel. Instructions, are found in the 911 Diagnostic Crash Procedures appendix in your *System Manager's Guide*.

Information Resources

- *R1000 Development System: System Manager's Guide*
- *Rational Environment User's Guide*
- *Rational Environment Basic Operations*

These course materials can be helpful reminders of the topics covered. Rational recommends that you make notes on your copy. At the end of these course notes are some appendixes containing useful information. Furthermore, some of the system documentation is particularly helpful for system manager.

- The *System Manager's Guide* can be your most valuable reference. It contains detailed information necessary for all system managers; it includes Series 200, 300, and 400.

  Supplementary information for system managers is included with many options Rational products, such as the Rational Design Facility and Rational Network Mail. These are generally packaged as tabbed set of pages that can be inserted in you System Manager's Guide.

- The *User's Guide* is a tutorial for beginners. System managers and other users who could not attend a Rational Fundamentals class will find this manual especially helpful; it acquaints users with the basics of getting started, editing, and simple Ada programming in the Rational Environment.

- The *Basic Operations* contains step-by-step instructions on how to accomplish simple tasks with the Environment. It also includes a section called "Basic Keymap", which lists and describes Environment commands that are bound to keys.

Information Resources (cont.)

- *Rational Environment Reference Manual* (11-volume set)
  - *Reference Summary* (RS)
  - *System Management Utilities* (SMU)

- The reference set describes all the Environment commands available on your R1000 system. While you may not need to consult this documentation daily, you should have a set available because it is the most complete, authoritative documentation about Environment commands and facilities.
  - The Reference Summary is Volume 1 of the *Reference Manual*. It contains an introduction to the documentation set, a list of all Environment commands, the "Parameter Value Conventions" section, a Rational glossary of terms, and a Master Index to all reference entries.
  - The System Management Utilities book is Volume 10 of the *Reference Manual*. It contains descriptions of most Environment commands related to monitoring and managing the R1000. The commands are grouped into Ada packages that are named according to the facilities they affect: Daemon, Message, Operator, Queue, Scheduler, System_Backup, System_Utilities, Tape, and Terminal. This volume has its own index where you can look up a command or process by name.

Review and Discussion

1. According to the descriptions in this introduction, what tasks might you be expected to perform that you do not yet know how to do?
2. Which course modules are most appropriate to your job role? least?
3. What documentation might you use daily as system manager? What documents might you use occasionally?
4. From what sources can you expect Rational support?
5. What Rational training should new users have first? What manual is a tutorial for beginners?

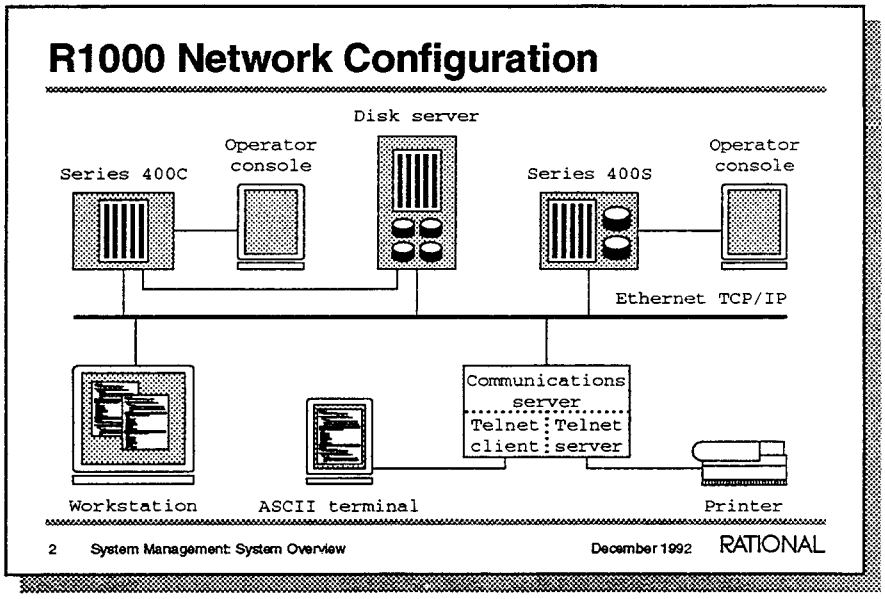
Answer these questions to review the points covered in this introduction. Some suggested responses follow.

1. Among the tasks mentioned are creating user accounts, making backups, and announcing the availability of new software tools.
2. For a system operator, the Operations module is most important and the Administration module is least important. For a system administrator, the importance is reversed.
3. The *System Manager's Guide* is probably the most important book for system-management personnel. Other useful documents include Basic Operations and the 11-volume Rational Environment Reference Manual.
4. Your local account team typically provides the most immediate support. The Rational Response Center can give telephone support.
5. The first Rational training course is Fundamentals. The *Rational Environment User's Guide* is a step-by-step learning manual that newcomers can use.

Course Modules

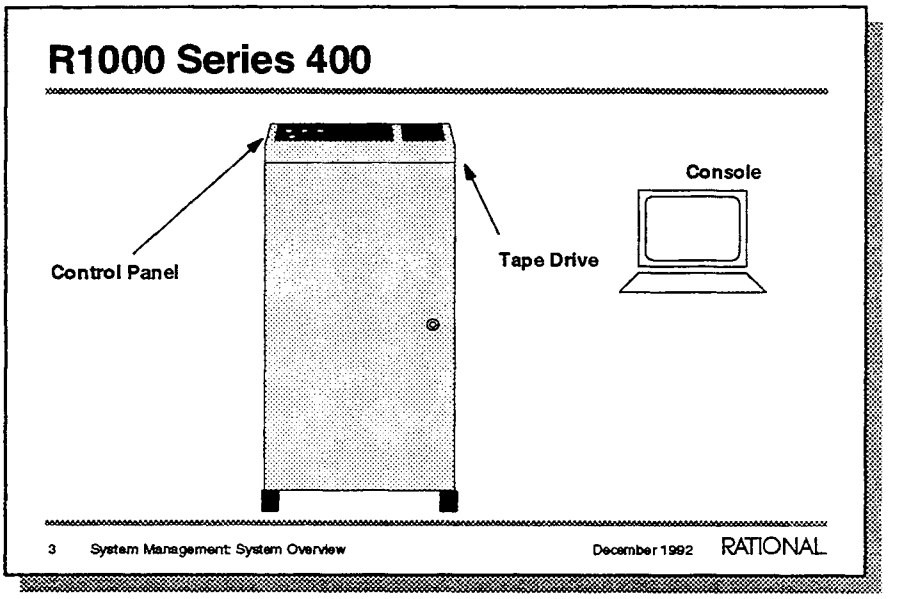
- Introduction
- **System Overview**
- System Operations
- System Administration

1 System Management: System Overview December 1992 RATIONAL



The Rational R1000 is designed to be used as a software-engineering server in a heterogeneous computing environment. Typically, one or many R1000's share a public Ethernet TCP/IP network with other devices.

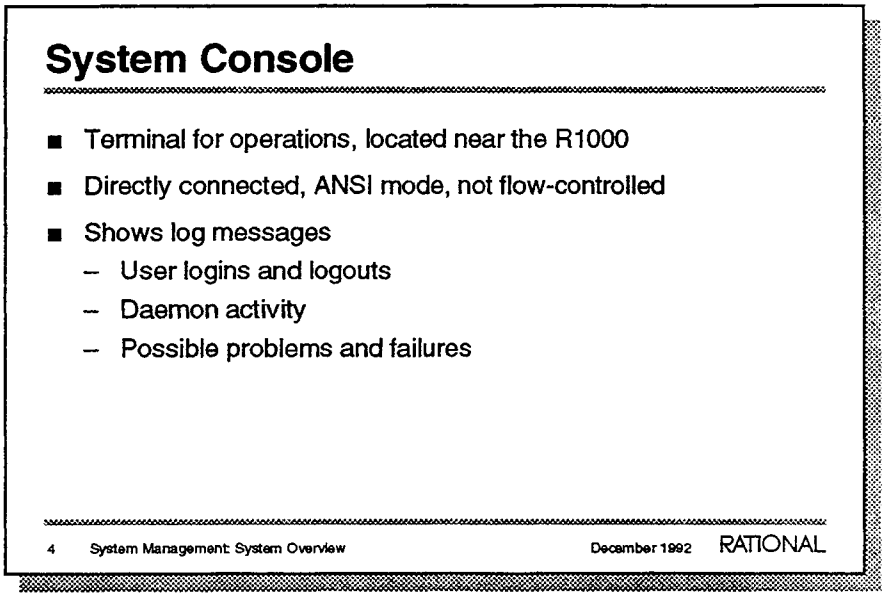
In its coprocessor configuration, the R1000 depends upon a separate disk/file server for disk storage. In this situation, the R1000 and the server share an additional private Ethernet UDP/IP network.



The operator-console is the major interface for R1000 system-operations tasks.

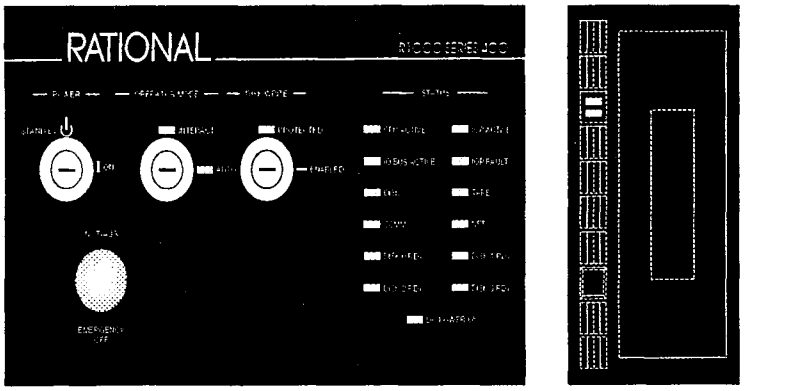
The control panel contains switches and indicator lights that affect or indicate system status.

The imbedded cartridge tape drive is the major I/O device for data backup.



- The console is typically in the climate-controlled computer room. This is helpful because system-boot, diagnostic, and tape-handling operations require access to system hardware.
- System software must always be able to communicate with the console. Some system messages may contain control characters that could be misinterpreted as flow-control characters. Therefore, the console must not expect flow-controlled messages.
- Log messages include indications of user logins by port number, routine messages from system-housekeeping software (daemon), and details about problems and errors—including a system crash. The console is the first place to look for indications of system problems.

Series 400 Control Panel



The control panel includes keyswitches, indicator lights, and the Emergency Off power button. The two-position keyswitches are locked when the supplied key is removed.

The POWER keyswitch enables delivery of AC power to the system's Power Distribution Unit (PDU). Set this keyswitch to STAND-BY to remove all power supplied to system components. Unless an emergency situation exists, power should not be removed from the system until an orderly shutdown has been performed.

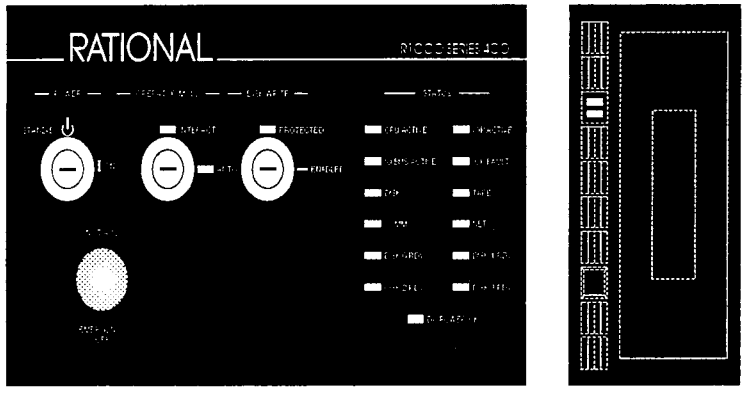
The OPERATOR MODE keyswitch controls system behavior upon boot. When this switch is set to INTERACT (such as, interactive mode), the firmware prompts for operator input following a system-reset or power-on sequence. When the switch is set to AUTO, the firmware operates according to predetermined options, which typically initialize most or all software without operator intervention.

The DISK WRITE keyswitch is normally in the ENABLED position so data can be written to the disks. During some maintenance operations and for initial system boot, you may place this keyswitch in the PROTECTED position. Note that if the write-protect switch is set to PROTECTED while the system is booted, it will crash.

Use the red EMERGENCY OFF button to remove power only in a real emergency, because loss of data may result.

(Status LEDs are described on the next page.)

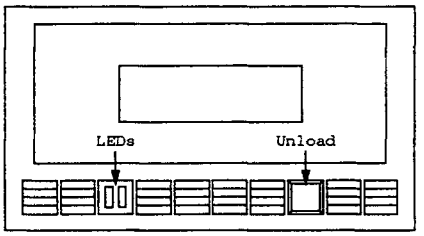
Series 400 Control Panel (cont.)



Status LEDs and their meaning

Label	Action	Meaning
CPU ACTIVE	Blinks	R1000 computer board is functioning
IOP ACTIVE	Blinks	I/O Processor board is functioning
IO BUS ACTIVE	Blinks	Data is passing along the internal bus
IO FAULT	Steady	Internal data error; suspected hardware failure
DISK	Blinks	Processing to/from disk is in progress
TAPE	Blinks	Processing to/from tape drive in progress
COMM	Blinks	Processing to/from serial ports in progress
NET	Blinks	Processing to/from Ethernet port in progress
DISK *n* RDY	Steady	Disk number *n* (0–3) is available
DC POWER OK	Steady	Internal DC power is functioning

8-mm Cartridge Tape Drive



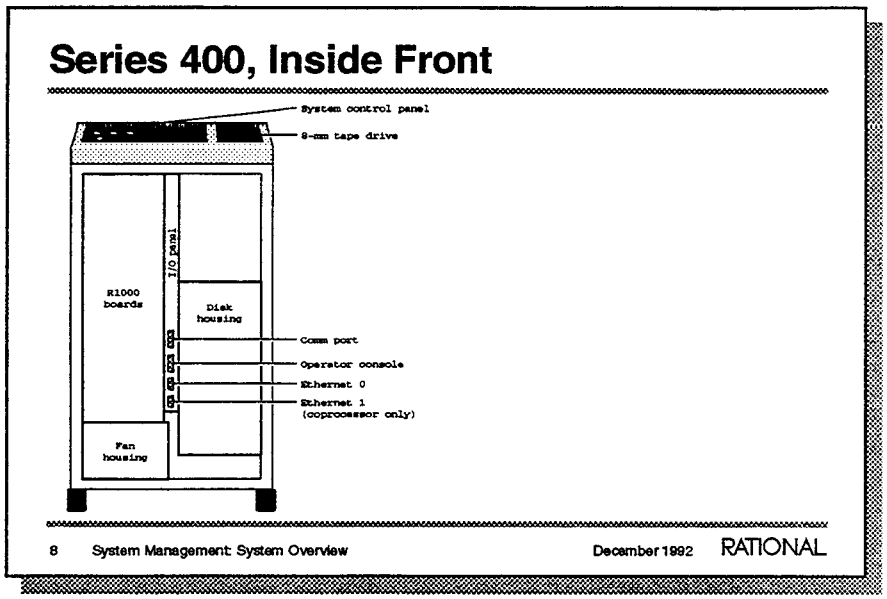
7 System Management: System Overview December 1992 RATIONAL

The tape drive provides data-backup capability. It is designed to use the same 8-mm magnetic tape cartridges that are used in some video cameras. See your System Manager's Guide for specifications and capacities of the tapes available. On many systems, a copy of all system data fits easily on one tape.

The Unload button opens the drive door, which is hinged at the edge next to the button. Closing the door with a cartridge in place starts the automatic load process; it is not necessary to do anything else to put the drive online.

The green LED lights when the tape is accessible. This may take a few minutes after the drive door is closed, depending upon the tape installed and the type of operation. The amber LED lights during tape I/O operations.

Following tape write/read operations, the tape is rewound automatically and the door opens automatically. Under normal circumstances (that is, unless the wrong tape is installed), operators should not need to use the Unload button to open the drive when a tape is installed.



This is the view you get when you open the front panel on your R1000 Series 400 system. The I/O panel and its connectors are shown in greater detail on a following slide.

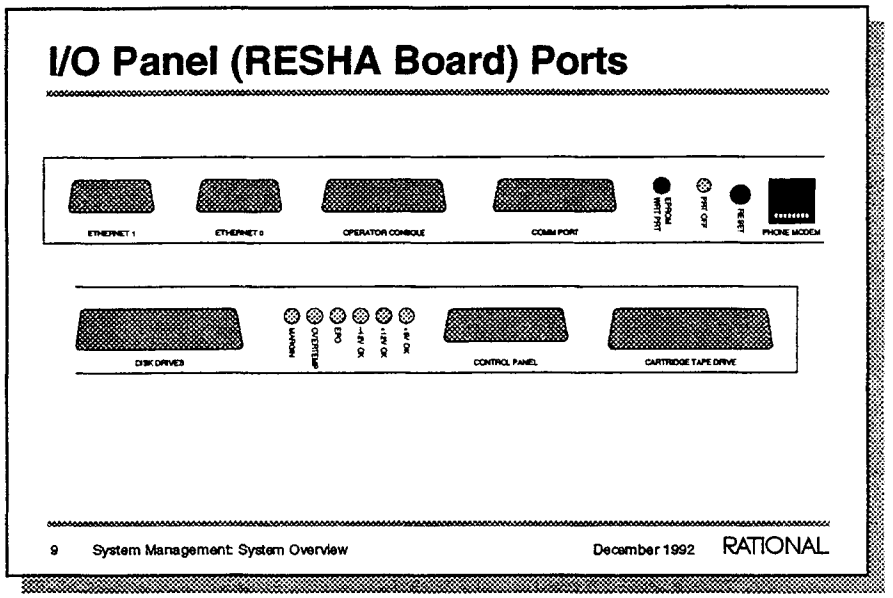
The other modules are enclosed. Generally, customer system management personnel will not need to open these internal enclosures; Rational support personnel may.

The "R1000 boards" module houses the main circuit boards for the system, enclosed between a backplane and a foreplane:

IOC	CPU, self-test PROMs, and board-interface logic
VAL	"Value board"
TYP	"Type board"
SEQ	"Sequencer board"
FIU	"Field Isolation Unit board"
MEM0	32 MB main memory
MEM2	(optional) additional 32 Mb memory board; replaces airflow-directing spacer

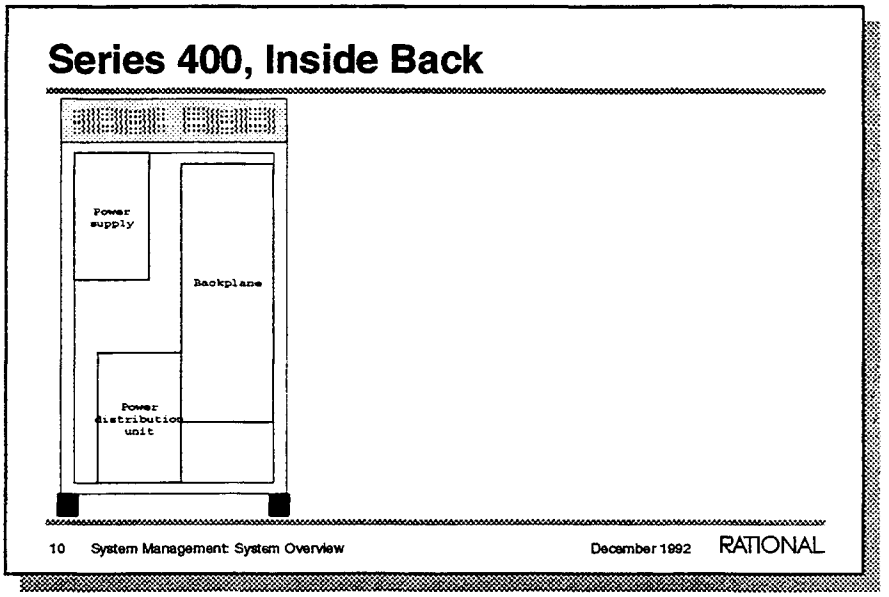
The "Disk housing" module encloses up to four disks, stacked vertically, each with a 1.1 Gb formatted capacity. The logical unit numbers for these disks are defined by their position: drive 0 is the topmost disk. Additional disks may be added at any time, up to the four-drive capacity. If your system is configured as a coprocessor (that is, using disk space on a network file server), it will have no disk drives.

The "Fan housing" module encloses the DC-powered cooling fans. These fans are on whenever power is applied to the system.



The RESHA (Rational Ethernet SCSI Host Adapter) board contains inter module and system-external interfaces with corresponding logic. Color-coded indicator LEDs may be consulted by system managers for quick information on system status: Red LEDs indicate a problem, Amber lights designate an unusual condition, and Green LEDs indicate normal operating mode. The connectors and indicators (from top to bottom) are:

- CARTRIDGE TAPE DRIVE — Internal SCSI interface connector
- CONTROL PANEL — Internal connector to control panel
- +5V OK — Green LED on when conditions are normal
- +12V OK — Green LED on when conditions are normal
- -12V OK — Green LED on when conditions are normal
- EPO — Red LED on when EPO button was pressed
- OVERTEMP — Red LED on when this problem occurs
- MARGIN — Amber LED on for abnormal power/timing margins
- DISK DRIVES — Internal SCSI interface connector
- PHONE MODEM — RJ45 (8-wire) telephone interface for auto dial-out/dial in
- RESET — Recessed "white button" for resetting system
- PRT OFF — Amber LED on when IOC EEPROM is writeable
- EEPROM WRT PRT — Green LED on when IOC firmware is write-protected
- COMM PORT — External DB25 Asynchronous interface for Port 16
- OPERATOR CONSOLE — External DB25 Asynchronous interface
- ETHERNET 0 — External interface for TCP/IP, public Ethernet
- ETHERNET 1 — External interface for UDP/IP, private coprocessor Ethernet



System managers may need to access the Power Distribution Unit (PDU) module to inspect circuit breakers or the 250V 25A fuse. To gain access, go through the back panel of the R1000 and open the PDU module cover.

The Operations module of this course will discuss more about the protection afforded by the circuit breakers in case of power problems.

Course Modules

- Introduction
- System Overview
- **System Operations**
- System Administration

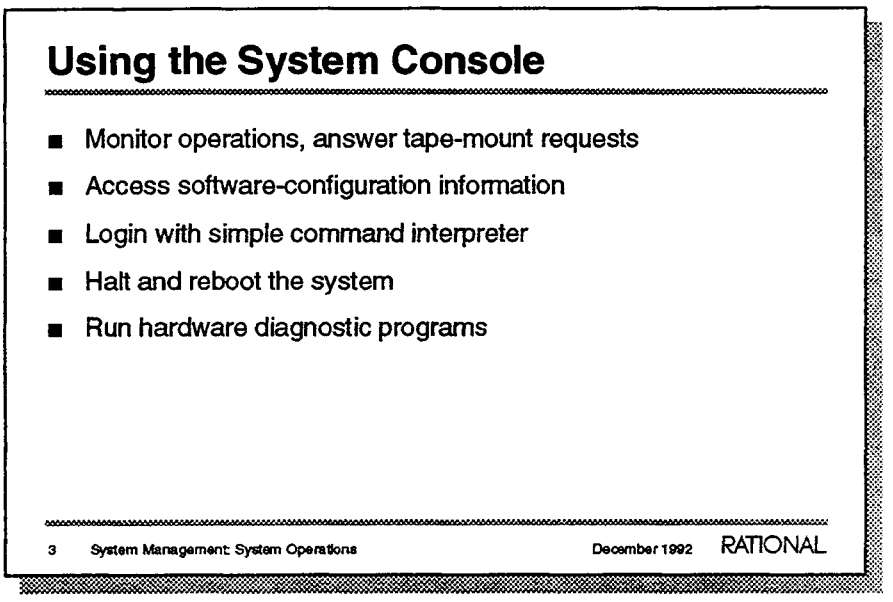
1 System Management: System Operations December 1992 RATIONAL

Operations Activities

- Monitoring system
- Responding to tape-mount requests
- Preserving system data with Backup and Archive operations
- Stopping and restarting the system
  - Under normal circumstances
  - In emergency situations

These are the primary activities of R1000 operations personnel. In this module we will examine each of them in detail.

- Status messages and indications of system functions are displayed routinely on the system console. Indications of problems are also shown there.
- User- and operator-initiated actions that involve reading or writing magnetic tape result in console messages requesting operator action.
- Operators typically have responsibility for regular and on-demand backup of system data.
- Orderly and/or emergency shutdown of the system, investigation of possible hardware problems, and system reboot are typically handled by operations personnel from the system console.



The console is the operations interface to the R1000. All system-level functions are designed to be executed from this device.

- Operators can observe system activity through the console. User-initiated tape-mount requests are communications that require operator action.
- Low-level software accessible from the console (EEDB and Kernel) can provide information about the system-software components.
- You can log in to the Environment from the console and execute many simple commands. However, display- and window-oriented commands are not effective because the console is assumed to be a simple command-oriented device.
- The system console is the recommended interface to use when shutting down the system. It is the required interface for booting the system.
- In case of suspected hardware problems, you or Rational support personnel may run ROM-based diagnostic software from the console, even when the system is not fully functional.

Steady-State Operations

- Messages are logged on the console
- Various command interfaces ([Control] [Z] cycles between them)
  - Elaborator Database (prompt EEDB:)
  - Kernel (prompt Kernel:)
  - Command interpreter (prompt Username: or Command:)
- Tape requests

- During normal operations, information messages indicate user login and logout activity, network transmission retries, and other normal occurrences.
- The active interface is indicated by a message and a prompt on the console that is similar to:

```
====>>  Kernel.11.5.0  <<====

Kernel:
```

- Tape-mount requests appear on the console when a tape operation is requested. They ask the operator to mount a magnetic tape in the drive. This is one reason for having the console in the computer room, adjacent to the system.

EEDB Interface

- For displaying or changing configuration of system software
- Display commands:
  - Show_Default
  - Running
  - Vdisplay configuration

- In certain situations, Rational support personnel may ask you to view the Environment configuration; for example, you may need to know the version number associated with a system-software component when you report a possible bug. You will not be changing the configuration yourself, but your Rational representative may do this when installing updated software.

-
  - The Show_Default command displays the name of the default configuration (for example, D_12_2_4) on the console.
  - The Running command displays the names of the system components and the configuration currently running. Sample output:

```
ED (partial)
DDC
OM (partial)
TCP_IP_TEST (partial)
DT (partial)
D_12_2_4
```

  - The Vdisplay command lists all software components in the configuration (about 40 lines), including version numbers and creation dates.

EEDB Interface (cont.)

- Action commands
  - Snapshot
  - Elaborate
  - Unelaborate
- Enter kernel (command Kernel, use Quit to return)

- The Snapshot command causes the system to commit changed images to disk. This is usually scheduled to run periodically and automatically (for example, every 30 minutes), but the action can be run explicitly with this command.

  The Elaborate and Unelaborate commands can change the system-software configuration by installing or deinstalling subsystems (components). You should only execute these commands under the direction of Rational support personnel.

- The Kernel command lets you address the lower-level Kernel software from within the EEDB. The Kernel layer can provide information about individual user and system jobs. Use the command Quit at the `Kernel:` prompt to return to the EEDB. This method of accessing the Kernel is preferred over access via [Control] [Z] because incorrect commands sent directly to the Kernel (that is, not through the EEDB) can crash the system.

Exercise: Using EEDB Interface

1. Execute the Show_Default and Running commands.
2. Using the name of the default configuration (for example, D_12_2_4), execute the Vdisplay command.
3. Initiate a snapshot.
4. Go to the `Kernel:` prompt from within the EEDB and return.

1. Observe the displays.
2. The command is:

```
EEDB: vdisplay D_12_2_4
```

3. The command is:

```
EEDB: snapshot
```

   Notice the log messages indicating progress of this action.

4. The sequence of commands is:

```
EEDB:  kernel
Kernel:  quit
EEDB:
```

Kernel Interface

- Provides information about:
  - Jobs and job scheduling
  - Disk-volume utilization and garbage collection
  - I/O ports
  - Error logging

The Kernel provides very low-level system commands without a lot of protection against operator errors. Rational support personnel may suggest that you run Kernel commands.

The commands described here are common ones that you may use for information. Other Kernel commands can change the operating environment of your system.

Kernel: Jobs Command

- Sample output

```
  Job Pri Stat CPU%  ModCt Cache   Disk  PgLim DskWts D/S JSegSz WsSiz WsLim
  --- --- ---- ---- ------ ----- ------ ------ ------ --- ------ ----- -----
    4   0 R,AT    4   4305  8968  16948  65536 005908   0   3614 11004 11000
    5   0 R,AT    0     12    62     73  65536   5343   0    138    25   200
  248   6 I,CE    0     49   177    294  16000   1809   0     68   128   150
  249   0 I,DT    0      9    27      5   8000     88   0     52    77   100
  250   6 I,AT   13      1     0      4   8000     73   0     72     0
  251   6 I,OE    0      1   100      1  16000    327   0      7   141    75
  252   6 I,CE    8     49   133    266  16000   1970   0     59   128   150
  253   6 I,CE    0      1     2      5  16000    280   0      3     3   150
  254   0 I,SV    0     13     4     37   8000    162   0     40    26    75
  255   6 I,CE    0      1     7      6  16000    154   0     14    29    10
```


Column	Definition
Job	Job number (Job 4 is always the Environment, 5 is system daemon)
Pri	Priority
Stat	Status (see symbol lists below)
CPU%	Percentage of CPU used
ModCt	Number of packages and tasks
Cache	Number of pages in cache
Disk	Number of pages on disk
PgLim	Page limit for this job
DskWts	Total disk waits (roughly, page faults)
D/S	Disk waits per second
JSegSz	Job segment size
WsSiz	Working set size
WsLim	Working set limit

Job Status symbols:

R Running I Idle Q Queued W Withheld D Disabled

Job Kind symbols:

CE Core editor OE Object editor SV Server AT Attached job DT Detach

Kernel: Other Job Commands

- JOB_NAMES gives segment numbers
- SHOW_TASK_STATES displays status
- DISABLE_JOB
- ENABLE_JOB to reenable

- Segment numbers may be useful in support or diagnostic situations.
- Using the Show_Task_States command to display status can help with diagnosis of suspected problems.
- If a runaway job is consuming system resources, you may be instructed to use the Disable_Job command. Use it with care, because disabling some system jobs (for example, a core editor) can halt user sessions and result in loss of data.
- The Environment equivalent is the Job.Enable command.

Kernel: Scheduler Commands

- SHOW_MTS_PARAMS displays settings for the medium-term scheduler
- MTSQ displays the medium-term scheduler's queues
- LOAD displays the load factors for all scheduler queues

- The Medium Term Scheduler is a system process that manages jobs.
- Mtsq command sample output:

```
Foreground Q
203
219
192
231
140
Background Q
169
151
Internal Transition Q
Job 140 => #7708C
```

- Load command sample output:

```
Run Queue Load      => 4.07, 2.05, 1.95, 1.67
Disk Wait Load      => 0.00, 0.50, 0.50, 0.58
Withheld Task Load  => 0.00, 0.10, 0.12, 0.10
Available Memory    => 8613 pages
```

Kernel: Disk-Information Commands

- SHOW_VOLUME SUMMARY displays disk-usage and garbage-collection thresholds
- CHANGE_GC_THRESHOLDS changes garbage-collection thresholds
- SHOW_GC_STATE displays current status

- Show_Volume_Summary command sample output:

```
Volume Status Summary
Vol     Total       Unused       Rate
Num    Capacity    Capacity    Blks/Min
-------------------------------------
 1     1077975      743502          0
 2     1109700      755200          0
 3     1109700      798804          0

low space thresholds for volume 1:
  START_COLLECTION threshold at 25% (waiters exist)
  RAISE_PRIORITY threshold at 15% (waiters exist)
  STOP_JOBS threshold at 12% (waiters exist)
  SUSPEND_SYSTEM threshold at 7% (waiters exist)
  SPACE_04 threshold at 0% (no waiters)
  next trigger at 269494 blocks
 ....
```

- Consult your Rational representative before changing the garbage-collection thresholds of the disks.
- This indicates the state of the disk daemon (garbage collection) if it is running; otherwise, indicates that it is not running.

Kernel: Miscellaneous Commands

- SHOW_ERROR_LOG displays recent logs
- SHOW_PORT_INFO displays I/O traffic for a port
- TIME gives current system date and time
- SHOW_CONFIGURATION_BITS displays selected configuration options

- You are prompted to identify the number of log entries to view. Typically, about 50 entries fit on the console screen.
- This display indicates the number of bytes and packets transmitted through this port.
- The Time command displays the date and time.
- These include the keyswitch settings and current values for other configuration choices, as described later under "Boot Options."

Exercise: Accessing the Kernel

1. On the operator console, enter the Kernel from the EEDB interface.
2. Try some information commands, such as JOBS, LOAD, MTSQ, SHOW_GC_STATE, SHOW_ERROR_LOG, and observe the output.
3. Exit back to the `EEDB:` prompt and view the current configuration with the command Running.

1. The command is:

```
EEDB: Kernel
```

2. Examine the output. You need not memorize abbreviations, but you should be able to interpret most of the data.
3. Use the Quit command to get from `Kernel:` prompt to the `EEDB:` prompt.

Console Command Interpreter

- Permits logging in to Environment from console
  - Can execute many Environment commands, but not the same as normal Environment login
  - Times out after two minutes of inactivity
- Useful commands include
  - Full_Backup; for initiating backup activities
  - Users; for a list of current user sessions

For more about the Console Command Interpreter (CCI) including descriptions of typical commands, see the "Operator Console Command Interpreter" appendix in your *System Manager's Guide.*

- Sample login session:

```
=====>> Console Command Interpreter <<=====
Username: Operator
Password:
Session: tom
08/03/31 08:52:22 --- operator.tom logging in.

=====>> Ci.Interpret (OPERATOR.TOM Job 157) <<=====
command:
```

  The Environment's Format, Semanticize, and Promote operations that are associated with Ada programming are not available through the CCI.

  –

  – The prompt changes from command: to Username: when your session terminates.

- Commands may be entered at the prompt. These use Ada language syntax, so punctuation should be exactly as shown. Terminate each command with a semicolon (;) and [Return].

  –

  –

Exercise: Using the CCI

1. Login to the Environment as user Operator.
2. Execute the command Users; and observe the display.
3. Execute the command
   Typ("!Machine.Editor_Data.Daily_Message");
4. Execute the command Quit to exit.

1. A typical convention is to use your username as a session name. Note that the password does not echo on the console.
2. Type exactly the characters within the single-quotes and press [Return]. Look for your session in the output.
3. Message for all users is held in this text file and displayed when users login.
4. You may end your session with this command. You may also let the session terminate by not entering a command for two minutes.

Handling Tape Requests

- Generated by Backup, Archive, Tape.Read, and Tape.Write operations
- Prompts require a response

```
====> ANDERSON.S_1 % Tape.Read <====
Please Load Tape
  (Kind of Tape     => STRAIGHT_ANSI,
   Direction        => READING,
   Volume Id        => 005020,
   Additional Info => get tape from pavel)
Is Tape mounted and ready to read labels?
```


For more information about tape operations, see chapter 6 "Tape Information and Operations" in your *System Manager's Guide.*

- Archive.Save, Archive.Restore, Tape.Read, and Tape.Write are user commands in the Environment that are typically used for tape I/O. The commands Full_Backup and Primary_Backup may be entered by the operator or system manager for periodic system backup operations.
- Values displayed in parentheses identify the operation and the tape to be used. The Additional_Info field contains special instructions, if any, for the operator; otherwise, the field is omitted.

  If you do not answer the prompt correctly, the program indicates what the expected responses are and the prompting question reappears.

Handling Tape Requests (cont.)

- Procedure: load tape, respond yes
- Tape-label information is displayed and additional prompt

```
OK to read Volume? [YES]
```

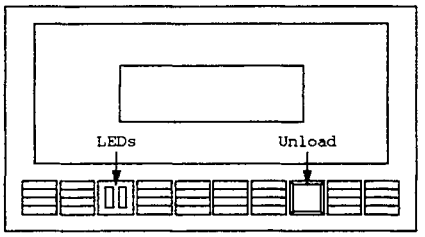
or

```
OK to overwrite Volume? [YES]
```

- After the operation, a dismount request is displayed

- Loading a cartridge tape means placing the tape in the drive and closing the door.
- The value in brackets [and] is the default response. Press [Return] to accept this value. There may be other prompts concerning assignment of a volume id for the tape.
- The tape-drive door opens when the tape can be removed. Do not attempt to open or close the door by force. The dismount request does not require a response at the console.

Cartridge-Tape Drive



For more information about the cartridge-tape drive, see chapter 6 "Standard Tape Drive" in your *System Manager's Guide.*

The left LED is amber; it lights during tape I/O operations. The right LED is green; it lights when a tape is loaded and ready.

Normally, the tape door will open when an operation is complete. You may press the unload button to halt an operation, then press again to open the door.

Common Tape Errors

- Tape is on not online
- Tape is unreadable
- Tape for write operation is write-protected
- Wrong drive

Things to check when a tape operation is not proceeding as expected:

- After the door is closed on a cartridge-tape drive, the tape is loaded. This may take several minutes. The green LED on the drive lights when the tape is accessible.
- There is no remedy for an unreadable tape. We recommend that you verify system-backup tapes when you make them.
- The write-protect tab for a cartridge tape is a plastic slide on the outer edge. The tape is write-protected when this slide is closed.
- Some R1000s can be equipped with two different tape drives (8-mm cartridge and 9-track). On such systems, you must specify the tape-drive number in a dialog for each mount request.

Exercise: Handling Tape Requests

1. Login to the Environment through the CCI and execute the command Archive.Save("@");
2. Place a tape in the system tape drive, but do not close the drive door
3. When the prompt asks if the tape is online,respond yes

   Observe the error message and new prompt
4. Close the tape-drive door and respond yes again

1. This command saves to tape all the objects in the current context. When you log in, the current context is your home library (for example, !Users.Operator).
2. The tape is not "mounted" unless the drive door is closed. We'll see how the system responds.
3. The mount-request software repeats the prompt. (You can also answer "no"; if you do, you'll be asked why you did not mount a tape. Enter "?" to see a list of valid replies. The reason will be relayed to the original requestor.)
4. After you close the door, the tape will take a few minutes to load. The green LED lights when the tape is ready. (The adjacent amber LED lights when the tape is written on or read.)

System Backups

- Kinds
  - Full records entire system state, usually done weekly
  - Primary records changes made since last Full, usually done daily
  - Secondary records changes made since last Primary, not used
- Impact system performance
- Backup log visible from Environment with System_Backup.History

For more information about backup operations, see chapter 7 "Saving and Restoring System Data" in your *System Manager's Guide.*

- System backups record information from the most recent snapshot. You should make a snapshot just before beginning a backup operation.

  Restoring information from a Primary backup assumes that data corresponding to the last Full backup are already present. Restoring information from a Secondary assumes that data from the Primary are already there.

- Typically, you'll plan backup operations when users are not on the system.
- You can execute this command from the CCI. When executed with default parameters (as shown), this command produces about 50 lines of output.

Making a Backup

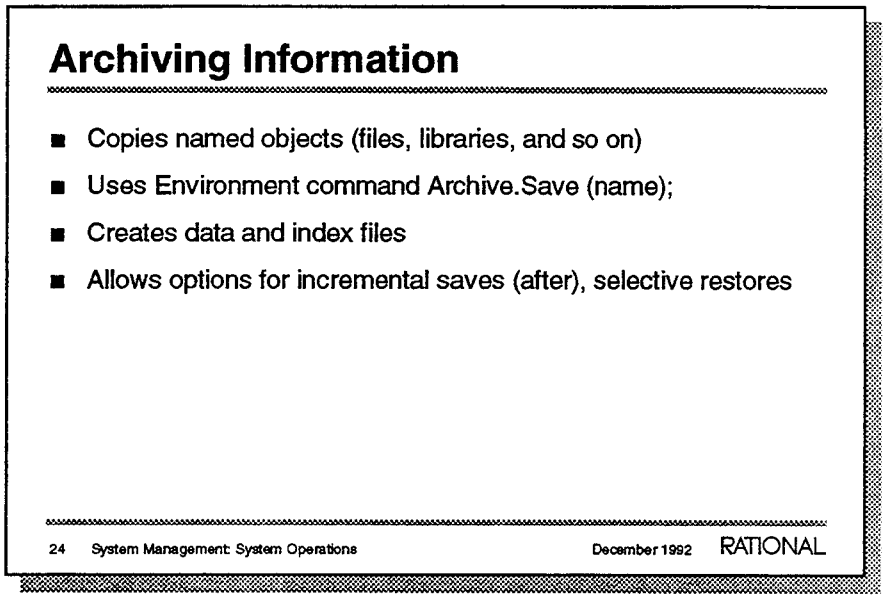
- Process
  1. Login as Operator
  2. Execute command Full_Backup; or Primary Backup;
  3. Respond to tape-mount request
  4. Verify tape contents
  5. Dismount and label tape when backup completes

Backups represent all the data and programs on a system. To restore from a backup, you must recreate the entire set of information; individual files or objects cannot be recovered. (Use Archive operations to prepare for this kind of data recovery.) When executing a Full_Backup, a starting time can be specified.

- These are the basic steps for making a backup. You may want to find out if users are on the system (for example, with command Users;) and send them a message (`Message.Send_All (Message => "backup starts in 5 minutes");`)

  so they are not surprised by a lack of system responsiveness while the backup is in progress.

  When executing a Full_Backup, a starting time can be specified.



Data saved with Archive operations is selectively restorable.

- Archive commands can be used to save information to tape, directory, or another machine. The default saves to tape.
- This command requests a tape mount unless the operation is directed to a directory on this machine or on another R1000.
- The files are not visible in archive operations to or from tape.
- For more information, see the documentation for package Archive in the Library Management (LM) book, of the Rational Environment Reference Manual.

Exercise: Saving/Restoring Data

1. Take a full system backup
2. Verify the backup
3. Restore the system

1. This operation usually takes two hours or more. (There may not be time for you to do this now.)
2. Restoring the system requires shutting down and temporarily changing some boot options. All data on the system will be overwritten. You may not be able to do this yet; your instructor may arrange for you to practice this later.

Archives will be discussed in more detail in the System Administration module.

For more information, see Chapter 7, "Saving and Restoring System Data," in the *System Manager's Guide*.

System Shutdown

- Normally, use Environment command Schedule_Shutdown
- Alternatively, save state with snapshot and use [Break] on console (menu option 0)

```
Please Enter:
     0 => Restart System
     1 => Ignore break key
     2 => Redisplay recent console output
Enter Option:
```


For more information about shutting down the R1000 system, see chapter 2 "Stopping and Starting the System" in your *System Manager's Guide*.

- For planned or nonemergency shutdowns, Rational recommends this command. For more information, see the next slide or the documentation for Operator.Shutdown in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.
- Using the console [Break] key to shut down and restart the system can cause unnecessary loss of data. Consider this option only when serious problems exist, and always try to take a snapshot from the `EEDB:` prompt first.

  Note option 2, which is helpful for redisplaying system error or warning messages displaced by normal log messages. To view recent activity safely, use option 2 after [Break]; this may result in the redisplay of 10–20 lines of output.

System Shutdown (cont.)

- Notify users in advance
  - Message.Send_All ("System going down, please log off");
  - Can use What.Jobs followed by Operator.Force_Logoff
- Begin shutdown with Schedule_Shutdown

System shutdown must be coordinated with the user community and system-management staff. Rebooting a heavily used R1000 can take over an hour, so a system shutdown can curb productivity. Planned shutdowns may be scheduled to minimize impact on users.

- Get users to commit their changes and log off, if possible. For more about these commands, attend the system-administration module and/or refer to the *System Manager's Guide*.

  –

  –

- The Schedule_Shutdown command performs a sequence of actions, including a few warnings to users and shutting down the system at a specified time of day. (The default time is 11:59 p.m.)

  Schedule_Shutdown is the same as first executing an Operator.Shutdown_Warning followed by an Operator.Shutdown.

Power Interruptions

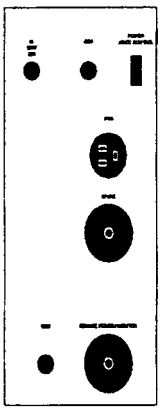
- Local power outage, planned or unplanned
- Power surge may trip circuit breakers
- Temperature too high
  - System shuts down automatically, logs messages
  - Restart system when suitable ambient temperature is achieved

During power outages, do what you can to ensure the integrity of system data and the security of the system components. If the system is still running, yet you notice power fluctuations, you may want to initiate a snapshot from the console, and begin an orderly shutdown.

In case of power loss, system data can only be as current as the last snapshot.

-
-
-
  -
  - In high-temperature conditions, warnings will appear on the console. The system will shut down automatically after a period of time (usually, 20 minutes).

Series 400 PDU



Check and reset circuit breakers when power conditions return to normal.

System Power-Off Procedures

- Emergency power-off:

  Press Emergency Off button on control panel and turn power keyswitch to Stand By position

- Normal power-off
  1. Shut down the Environment with 'Schedule_Shutdown'
  2. Write-protect the disks
  3. Turn power keyswitch to Stand By position

- The red Emergency Power Off button immediately removes power from all system components. Using this button generally causes the loss of some data or work.
- Using the disk-protect keyswitch on the system's front panel will write-protect the disks, further ensuring the integrity of their data.

Restarting after Power Off

- Check:
  - Power keyswitch in Stand By position; disks write-protected; circuit breaker reset
- When power-interrupt condition is fixed:
  1. Turn Power keyswitch to On.
  2. When DC Power OK and Disk Ready LEDs are on, write-enable the disks.
  3. Press the white button.
  4. Respond to console prompts and reboot the system.

For more information about starting your R1000 system, see chapter 2 "Stopping and Starting the System" in your *System Manager's Guide*.

- This is the expected condition of the system when the power is off.
- The white button (located on the edge of the I/O panel and labeled "RESET") resets the system-hardware components and initiates the boot process.

Boot Options

- When Operator Mode keyswitch is set to Interactive, system asks for values during boot sequence
- Options
  - Modem dial-out, modem answer
  - I/O processor autoboot, R1000 autoboot
  - Auto crash recovery
  - Console [break] key

- The Operator Mode keyswitch position is significant immediately following a system reset. To change the boot-crash-maintenance options, set the keyswitch to Interactive before the system reset occurs. Otherwise, the system boot will proceed automatically.
- Available options:

Option	Default	Meaning of default	Additional notes
Modem Dial-out	Y	Calls Rational support in case of hardware failure	Part of support agreement
Modem Answer	Y	Allows Rational to connect for remote debugging	Part of support agreement
I/O Processor Autoboot	Y	Allows IOP software to boot itself	If No, asks whether to boot from tape or disk
R1000 Autoboot	N	Asks what configuration to use, permits diagnostics	If Yes, boots from last identified default config.
Auto Crash Recovery	Y	Permits the system to attempt recovery automatically	If No, halts in IOP low-level debugger
Console [Break] Key	Y	Enables [Break] key for halting system, viewing messages	Recommended

Phases of Boot Process

- Run I/O diagnostics
- Boot IOP
- Query for configuration to boot
- Load Kernel

- This takes about a minute, and results in messages such as :

```
Testing I/O Adapter ...ok
Testing I/O Processor ...ok
```

- This also takes about a minute; the messages are:

```
Booting I/O Processor ...
IOP Bootstrap Version #.#
    ...
Restarting the system after ...
```

- For example:

```
Enter name of configuration to boot [Standard] :
```

  Press [Return] to take the default response.

- The Kernel layer, which presents the `Kernel:` prompt, is loaded next.

Phases of Boot Process (cont.)

- Start virtual memory
- Elaborate EEDB
- Elaborate Environment
- Execute initialization routines
  - System
  - Site-specific

- The output may be

```
====>>  CONFIGURATOR  <<====
starting diagnosis of configuration
starting virtual memory system
```

  This can take 10–40 minutes, depending upon how much data is on disk.

- The lower levels of the Elaborator Database must be installed (elaborated); then the EEDB: prompt appears while other software is installed.
- Sample output:

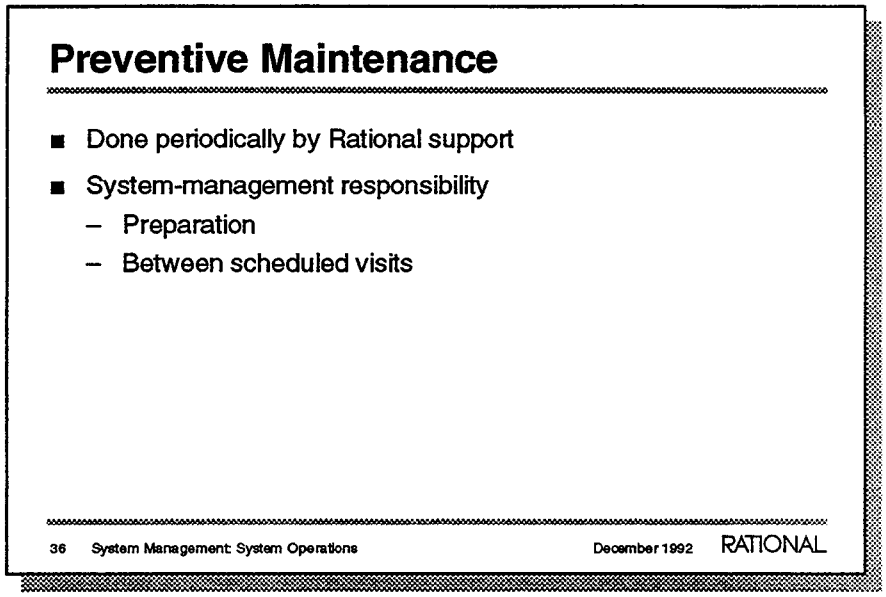
```
CMVC.11.5.2
FTP_INTERFACE.11.0.3
ARCHIVE.11.2.6
   ...
```

- Local initialization routines can be provided to establish printer queues, network servers, or other site-specific software on the R1000. See the release note for Delta 2.2, which can be found online in !Machine.Release.Release_Notes, for more about these.

Exercise: Halting and Restarting the System

1. Use Schedule_Shutdown to halt your R1000 system.
2. Reset the system with the "white button."
3. Reboot using the standard configuration. Note console messages that identify the boot phases.
4. Change the Operator Mode keyswitch to Interactive.
5. Use the [Break] key to reboot the system. When you are queried, change the R1000 Autoboot option to Yes.

Time and circumstances may not permit you to perform this exercise.

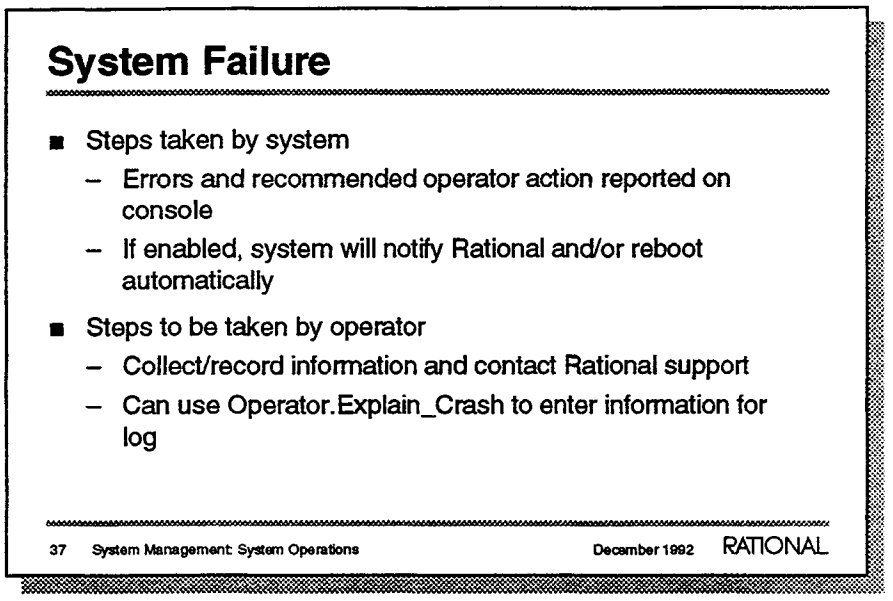


Preventive maintenance can identify, isolate, and correct potential system problems before they can adversely affect system performance or availability.

- Preventive maintenance (PM) visits are part of your Rational support contract. Visits are arranged well in advance.
-
  - System managers should prepare the user community for machine shutdown (for several hours) and perform a full backup.

    Rational support personnel will inspect the equipment, run test routines, collect and print reports, and discuss local machine-usage and operations policies with system-management personnel.
  - Between PM visits, system managers should alert Rational support personnel of system crashes, unplanned power interruptions, unusual disk collection, or system daemon activity.

    The tape drive should be cleaned with the Rational-supplied cleaning kit, or its equivalent, after every 30 Gb of data read/written or once a month, whichever comes first. The cleaning cartridge should be run through the tape drive two consecutive times to complete a monthly cleaning.



For more information about dealing with system failures, see the "Diagnostic Crash Procedures" appendix in your *System Manager's Guide*. This includes detailed directions for making a crash dump tape, running diagnostic programs, and other recommended operator actions.

- Messages on the console will indicate what has happened, what action the system has taken, and what is expected of the operator. For example, you may be asked to make a crash-dump tape that can be analyzed by Rational specialists.
- In case of a system failure (crash), you should be prepared to collect information and work with Rational support personnel to identify and correct the problem(s).

Failure Diagnosis

- For suspected hardware problem
  - Execute FRU diagnostic programs, if so directed
  - Run disk exerciser, if so directed
- For suspected software problem
  - Make a crash-dump tape, if so directed
  - Determine whether system is hung or jobs are suspended

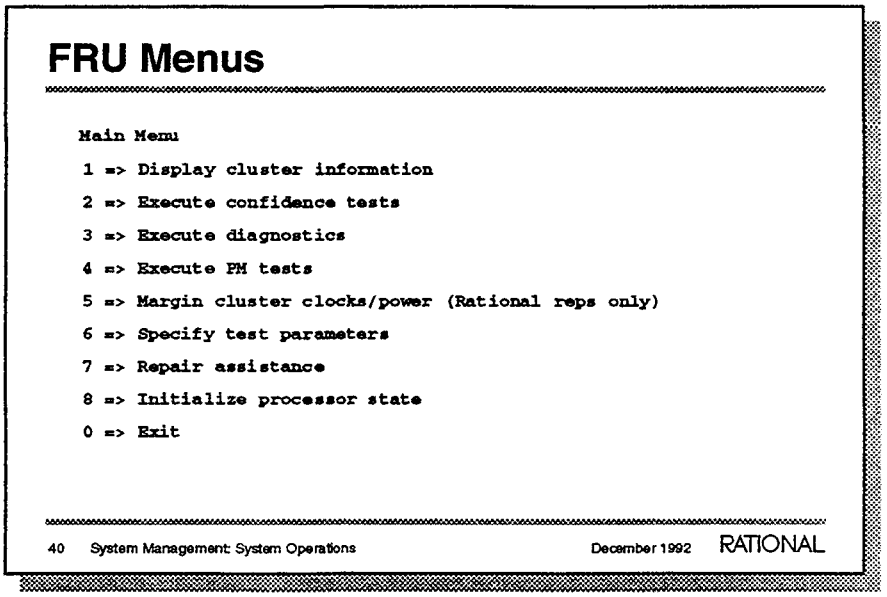
You should examine the "Diagnostic Crash Procedures" appendix in your *System Manager's Guide* and discuss the detailed procedures described with your local Rational technical representative.

- The system includes low-level diagnostic programs for the Field Replaceable Units, such as individual circuit boards. Generally, Rational support personnel will be available to guide you in using this software or the disk exercisers, which verify proper operation of these components.
- If the system has not crashed completely, use console-based commands to collect information about what system and user jobs are/were running.

Hardware Diagnostics

- Run at direction of your Rational support representative
- Executed from low-level CLI when Environment is not running; get prompt `CLI>` with [Control] [C]
- Execute command is abbreviated `x`
- Access field-replaceable unit (FRU) diagnostics with `CLI> x FRU`

- Crash-dump tapes may be the best way for Rational support personnel to diagnose your system's problems.
- The CLI prompt is accessible when the system is not running (such as immediately after the IOP has booted); press [Control] [C] to get the prompt `CLI>`.
-
- These diagnostic programs will be discussed in detail.



This is the main menu for the FRU diagnostics.

FRU Menus (cont.)

```
Diagnostics Execution Menu              (Main Menu Option 3)
 1 => Test the foreplane
 2 => Run all tests
 3 => Run all tests applicable to a given FRU
 4 => Run a specific test
 0 => Return to main menu
FRU Menu                                (Diagnostics Menu 3)
 1 => All                  7 => FIU
 2 => I/O adaptor          8 => Memory 0
 3 => SYSBUS               9 => Memory 1
 4 => Val ALU             10 => Memory 2
 5 => Typ ALU             11 => Memory 3
 6 => Sequencer           12 => Foreplane
```


These are examples of submenus for FRU diagnostics.

Other CLI Commands

- Disk exerciser: `CLI> x diskx`
- Initiate boot: `CLI> boot`
- Make a crash-dump tape: `CLI> crashdump`
- Display the shutdown (crash) log: `CLI> log`

Usually, you will run CLI commands only when diagnosing severe problems, and only at the direction of Rational support personnel.

Exercise: Running Diagnostics

1. Execute all FRU diagnostic tests.
2. Execute the disk exerciser.
3. Boot the Environment from the CLI.

To perform these exercises you may need to halt the system. If time and conditions permit, do so in the appropriate way: if there are users on the system, let them know and use Schedule_Shutdown; otherwise, you can use the [Break] key.

Course Modules

- Introduction
- System Overview
- System Operations
- **System Administration**

1 System Management: System Administration December 1992 RATIONAL

System Daemons

- Consists of background routines that manage and compact disk usage
- Contains several clients
  - Daily client: Consists of object-compaction routines that should be run once each evening
  - Disk client: Collects garbage generated by normal system usage
  - Error_Log client: Periodically updates the Environment error log
  - Snapshot client: Permanently saves updated objects
  - Weekly client: Consists of other object-compaction routines

-
-
  -
  - The Disk client the most important of the Daily clients.
  - The system manager should check the error log periodically for indications of problems.
  -
  -

Disk Client

- The Disk client collects garbage generated during the day
- The Disk client runs at various priority levels if the disks become too full
  - Start_Collection: Begins collection in background (yields to user jobs)
  - Raise_Priority: Competes with user jobs
  - Stop_Jobs: Stops all user jobs to complete collection
  - Suspend_System: Halts system

-
-
  -
  -
  -
  - This is an extremely serious condition. You should contact Rational before proceeding. Chapter 3 "Managing User Accounts" in your *System Manager's Guide*, provides steps on recovering from a Suspend_System condition caused by the Disk client.

Disk Client (cont.)

- Space thresholds can be adjusted
  - Example:

Threshold	Percentage
Start_Collection	25%
Raise_Priority	15%
Stop_Jobs	12%
Suspend_System	7%

- Can be set at boot time

-
  - Threshold setting can be displayed from the KEDB: prompt at the operator's console as follows:
    - Press [Control] [Z] until the KEDB: prompt is display.
    - Enter Kernel and press [Return]
    - Enter Show_Volume_Summary and press [Return]

    Consult with Rational before changing the thresholds.
- To set the thresholds each time the system boots, create a procedure called Local_Gc_Thresholds in the !Machine.Initialization.Local directory, following the text file example stored there.

  The recommended minimum levels fora series 400 are:

Threshold	Boot Volume	Other Volumes
Start Collection	12%	10%
Raise_Priority	10%	8%
Stop_Jobs	9%	6%
Suspend_System	7%	4%

  The minimums are higher for series 200 and 300 machines. Check with your local Rational representative before lowering the defaults on series 200 and 300 machines.

Disk Client (cont.)

- Space is not returned until the Disk client finishes
  - Rebooting the system before the Disk client finishes will *Not* recover any space
  - If the Stop_Jobs threshold is met, wait for the client to complete
  - If the Suspend_System threshold is met, follow instructions in Chapter 3 of the *System Manager's Guide*
- Some system-owned garbage is not recovered by the Disk client
  - System must be recycled periodically to recover this space
  - Once every week is recommended

-
  -
  - The system manager can also boot to DDC, but this should not be attempted without help from Rational.
  - If the Suspend_System threshold is met you should contact Rational before proceeding.
- This includes code segments. If the development team uses a lot of promotes and demotes, it may be necessary to recycle more often.
  - Periodic recycling is necessary for the continued functioning of the system. The Software_Catalog contains a server for rebooting the system on a specified scheduled.
  - A default of weekly recycling is recommended.

Snapshot Client

- The Snapshot client permanently saves system state
  - All saved objects are written permanently onto disk
  - Edited objects that are not saved are not written
- After system failure, the system reboots from the last snapshot
  - Changes made after the last snapshot are lost

-
  - An object is saved when you press [Return].
  -
-
  -

Client Scheduling

- An initial schedule is set up by Rational on installation
  - System manager may want to adjust the default schedule
  - Complete discussion appears in Chapter 5, "Maintaining System Efficiency" of the *System Manager's Guide*.
- Command: Daemon.Schedule

  Important parameters:
  - Client: Specifies the name of the client to schedule
  - Interval: Specifies the duration (in seconds) between runs
  - First_Run: Specifies the duration (in seconds) before the first run

7 System Management: System Administration December 1992 RATIONAL

Client Scheduling (cont.)

- Examples:

```
Daemon.Schedule (Client => "Snapshot",
                 Interval => 30 * Time_Utilities.Minute,
                 First_Run => 15 * Time_Utilities.Minute);
Daemon.Schedule (Client => "Daily",
                 Interval => 24 * Time_Utilities.Hour,
                 First_Run =>
                    Time_Utilities.Duration_Until_Next(2, 0, 0));
```

- Standard schedule
  - Snapshots are scheduled every 30 minutes
  - Daily client begins at 2:00 a.m.
  - Schedule is defined by machine initialization

-
-
  - Schedule definition will be discussed in more detail later. When the system boots, it automatically sets the standard schedule.
  -
  -

Daemon Control Commands

- Daemon.Run: Runs the specified client immediately
  - Example: `Daemon.Run (Client => "Snapshot");`
- Daemon.Quiesce: Delays the scheduled execution time

  Important parameters:
  - Client: Specifies name of client to delay
  - Additional_Delay: Specifies seconds to delay (default is one day)
  - Example: Skip garbage collection

```
Daemon.Quiesce (Client => "Disk", Additional_Delay => 86_400.0);
```

- Daemon.Status: Displays the status of all daemons

- You may want to force either a snapshot or disk collection to begin.
  -
- Daemon.Quiesce can be used to delay disk collection. Once the directory and the disk daemons have begun, you cannot stop them.
  -
  -
  -
-

System Data

- Managed by system managers
  - Modify files defining system configuration
  - Interrogate system-generated information
  - Set access control
- Located in !Machine

10 System Management: System Administration December 1992 RATIONAL

System Data (cont.)

- Accounting directory:
  - Contains system-accounting files
  - Contains Activity_@ files created with each boot
  - Accounting is kept if Enabled file exists when the machine boots
  - Information is available using the Accounting_Report command
- Devices contains:
  - Objects corresponding to all physical devices

-
  -
  - The completion of the filename contains the date of the boot.
  -
  -
-
  - Tools are available to read the attributes of these devices. This area should *Never* be modified by users or the system manager.

System Data (cont.)

- Editor_Data contains:
  - Default keymaps for all terminal
  - Help data
  - Daily message
- Error_Logs contains:
  - System logs created by Error_Log client
    Format: Log_YR_MO_DA_At_HR_MN_SC
- Groups
  - Used by access control

-
  - These can be edited to modify the system default keymap for a particular terminal.
  -
  -
- Examine using the reporting routines described later for errors and bad trends.
  - A user log is also created for each session with a filename format of <Username>_<Session_Name>. These are useful for diagnosing session-specific problems.
-
  -

System Data (cont.)

- Initialization procedures
- Machine_Name file
- SIMS: Contains problem-reporting database
- Search_Lists directory
  - Contains user searchlist objects
  - Default: Initial searchlist for new users
- Temporary directory
- Users directory
  - Contains users' account information

- These are system-tailoring procedures that execute at boot time. They will be discussed in more detail later.
- This file contains the name of the machine displayed in the Message window. This file should be edited to change the name.
-
-
  -
  - This can be edited to change the initial searchlist when new users are created.
- This temporary system/user file area is used by archive, among others, and is purged on reboot. Only objects that can be deleted with Library package commands should be placed here, for example subsystems do not belong here.
-
  -

System Data (cont.)

- Network-information files
  - Tcp_Ip_Name_Server
  - Transport_Name_Map
  - Transport_Routes
- Access-control files
  - User_Acl_Suffix
  - User_Default_Acl_Suffix
  - Operator_Capability

-
  -
  - Example:

```
ax25 224 mighty_mouse m68k_standard
ax25 226 timely m68k_standard
tcp/ip 89.64.1.2 gypsy Rational
tcp/ip 89.64.1.3 bud Rational
tcp/ip 89.64.1.17 washburn Rational
tcp/ip 89.64.1.18 clem Rational
tcp/ip 89.64.1.27 sheila Rational
```

  -
- This is discussed in more detail later.
  -
  -
  -

System Reporting

- System_Report.Generate displays information on:
  - System availability
  - System usage
  - Disk usage
  - Device errors
  - Daemon operation
  - System outages

- Report can contain all information or report on any of the following:

```
-- Report generation from system availability information.
-- Provide a variety of reports.
type Report_Class is (Availability, -- Uptime/downtime by
                                       classes
          Usage,          -- Per half hour, # users, etc.
          Disk,           -- Used disk space each day
          Devices,        -- Disk, Mem, Tape, etc errors
          Daemons,        -- Daemon state sizes and times
          Outages,        -- System outages and reasons
          Trouble,        -- Potential trouble areas
          Advice,         -- Advice on cleaning things up
          Everything,     -- All reports
          Tape_Mounts); -- Tape processing for backups
```

The System_Report package is located in the subsystem !Tools.System_Availability.

Accounting Statistics

- Accounting_Report command
  - Generates an accounting summary from information in !Machine.Accounting
  - Allows specification of Users, as well as beginning and ending dates
  - Example:

```
Accounting_Report (From_Date => "92/08/01",
        To_Date => "92/08/31",
        For_User => "vnv, gbd",
        Accounting_Directory =>"!Machine.Accounting");
```


16 System Management: System Administration December 1992 RATIONAL

Accounting Statistics (cont.)

- Accounting_Report command (cont.)

```
Accounting Summary for Period  1-AUG-92 to 31-AUG-92.
Total 'Work' Time for Period :  176 hours ( 22 days )

   NAME     LOGINS  LOGIN_TIME  CPU_TIME  DISK_IO  JOBS_RUN  P_LOG  L_CPU
----------  ------  ----------  --------  -------  --------  -----  -----
GBD              42     7/18:12   6:26:35   705487      3958    106     37
VNV              26     3/20:12   3:24:20   268458      3808     52     19
              <Avg>       <Avg>     <Avg>     <Avg>     <Avg>  <Avg>  <Avg>
Total ( 2 )      34     5/19:12   4:55:28   486972      3883     79     28
```


Key to the headings in the accounting summary report

```
NAME        : User account name.
LOGINS      : Number of times user has logged in.
LOGIN_TIME  : Total amount of time user was logged in.
CPU_TIME    : Total amount of CPU time user has consumed.
DISK_IO     : Total number of disk I/O requests for user.
JOBS_RUN    : Total number of jobs run by user.
P_LOG       : Percent user was logged in of 'Total Work Hours'
L_CPU       : 'Load' CPU, weights user consumption of available
              CPU time from 'Total Work Hours'.
```

Allocation of Worlds to Volumes

- Each disk drive is labeled with a volume number
  - Volumes are labeled 1 to N
- Worlds are allocated to volumes
  - All objects within a world reside on the containing world's volume
  - Library.Create_World command has volume parameter:

```
Library.Create_World (Name => "", Kind => Library.World,
    Vol => Library.Nil, Model => "!Model.R1000",
    Response => "<PROFILE>");
```

  - Library.Nil specifies that the new world should be allocated to the volume with the most available space

-
  - Machines may have 1, 2, 3, or 4 disk drives. Most machines contain 2 or more drives.
-
  - Nested worlds have their own volume allocation. Although one world may be inside another, all objects reside on the volume of the nearest enclosing world.
  -
  - This means the volume with the highest available space at this time. Volume 0 also is used for this.

Allocation of Worlds to Volumes (cont)

- Subsystems also are allocated to volumes

```
Cmvc.Initial (System_Object => ">>SYSTEM OBJECT NAME<<",
              Working_View_Base_Name => "Rev1",
                  .
                  .
                  .
              Volume => 0,
              Response => "<PROFILE>");
```

  - Volume => 0 is the same as Library.Nil

- Views also are allocated to volumes. All CMVC commands (like Release) have volume parameters.
  - That is, it places the new subsystem on the volume with the highest available space.

Management of Space Distribution

- Allowing the system to allocate worlds generally works well
- A common mistake is creating several users, worlds, or subsystems at the same time
  - All are allocated to the same volume
  - All objects reside on the same volume when they are finally populated
- Planning the distribution of worlds may be required

-
- This may not be a mistake in all cases.
  - One volume gets all the worlds because it continues to have the highest availability.
  -
-

Space Availability

- Basic commands
  - Operator.Disk_Space (sample output):

Volume	Capacity	Available	Used	% Free
1	1077975	611512	466463	56
2	1109700	890246	219454	80
3	1109700	900585	209115	81
Total	3297375	2402343	895032	72

21 System Management: System Administration December 1992 RATIONAL

Space Availability (cont.)

- Basic commands (cont.)
  - Library.Space (sample output):

```
        Object Total
Vol     Size   Size      Object Name
--- ------ ------ ----------------------------------------
                      !USERS.MCSHAN
                       .MCSHAN
         532            .ADAPROBLEMS            (DIRECTORY)
 2        90            .ADV_TOOLS              (WORLD)
          70            .ITVE                   (DIRECTORY)
 3       557            .MAILBOX                (WORLD)
 2       114    1363    .MCSHAN                 (WORLD)
```

- Console also provides space commands

22 System Management: System Administration December 1992 RATIONAL

Exercise: Allocating Resources

1. Create a world on Volume 2.
2. Verify that the world is located on Volume 2; and then delete the world.
3. Check the available disk space on the system.
4. Display the status of all daemons.
5. Generate a system report.
6. Generate an accounting report.
7. Display the status and state of the scheduler.

23 System Management: System Administration December 1992 RATIONAL

Exercise: Scheduling the Daemon and Visiting Data Areas

The class should do this exercise as a group.

1. Run the Snapshot client.
2. Schedule the disk daemon for 1:00 a.m.
3. Verify that the correct scheduling has been performed.
4. Delay the disk daemon for two hours.

24 System Management: System Administration December 1992 RATIONAL

Exercise (cont.)

5. Visit the following system-data areas:
   - !Machine.Accounting
   - !Machine.Editor_Data
   - !Machine.Editor_Data.Daily_Message file
   - !Machine.Error_Logs
   - !Machine.Machine_Name file
   - !Machine.Temporary

5.

- Placing a file named Enabled in !Machine.Error_Logs will save the error logs to disk.
-
-
-
-
-

Initialization at Boot Time

D_12_5_0 and later:

- The !Machine.Initialization world contains all initialization, including the following objects:

```
Local    : Library (World);
Rational : Library (World);
Site     : Library (World);
Start    : Ada (Load_Proc);
```

- !Machine.Initialization.Start executes all procedure contained in, or referenced by, Local, Rational, and Site.

26 System Management: System Administration December 1992 RATIONAL

Initialization at Boot Time (cont.)

D_12_5_0 and later (cont.):

- !Machine.Initialization.Rational is reserved for use by Rational, and should not be modified.
  - Cleanup and compaction
  - Layered products such as RDF and networking
  - Servers
  - Daemons

27 System Management: System Administration December 1992 RATIONAL

Initialization at Boot Time (cont.)

D_12_5_0 and later (cont):

- !Machine.Initialization.Site should contain customer-written units that are common to two or more machines at a given location.
- !Machine.Initialization.Local contains customer-written units that are unique to a single machine.
- Site and Initialization will not be overwritten by upgrades to the Environment
  - Password policy
  - Reboot_Server

28 System Management: System Administration December 1992 RATIONAL

Initialization at Boot Time (cont.)

D_12_5_0 and later (cont):

- !Machine.Initialization.Site and !Machine.Initialization.Local may contain:
  - Ada procedures
  - _Start files
  - Configuration files
  - Text files that tell the Environment how to initialize layered products like RDF or CDF
- See the *Guide to Machine Initialization* for release D_12_5_0, or later, for more detailed information

29 System Management: System Administration December 1992 RATIONAL

Rehosting Software on the Environment

- Move file to R1000
  - Make multiple calls to FTP
  - Copy files from ANSI-labeled tapes: Tape.Read
- Convert the files into Ada units: Compilation.Parse
  - Use wildcard to specify all files
  - Use another library for the Directory parameter to avoid name conflicts
- Compile the software: Compilation.Make

-
  -
  - This can be more efficient for bulk transfers of many files.

    Note that the ASCII standard has a filename limit of 40 characters. If two objects have different names, but are identical for the first 40 characters, they will appear identical when truncated to 40 characters. When transferred to the Rational Environment, multiple files with the same name will result in multiple versions. One way of dealing with this is to set the retention count > 2 when reading the tape and parse both versions.
-
  -
  - All units including subunits will parse nicely.
- No compilation script is necessary!

Tool Building

- User-defined tools are written in Ada
  - Language consistency
  - Full power of Ada
  - Full power of the Environment for development of tools
- Three major forms
  - Scripts of commands
  - System-programming tools
  - Customization of Environment interfaces and products

- This is a very powerful approach. There is no conventional job-control language to learn.
  -
  -
  -
-
  - Rather than writing a script file, as on most systems, the Environment supports command combination through the writing of Ada programs.
  - These are tools written with system-programming interfaces commonly used to perform analysis of Ada programs or projects structures.
  - This can take many forms: instantiation of the Target Build Utility, customization of the design product, development of user-interface skins for CMVC, and so on.

I/O Facilities

- Package !Io.Io provides streamlined operations similar to package Text_Io
  - Append operation
  - Standard error file in addition to standard input and output files
  - Preinstantiated Boolean, Integer, and Float I/O operations
  - Functional form of Get_Line

- The package !Io.Io offers higher performance than Text_Io. It does not have enumeration I/O, or column, line, or page control.
  -
  -
  -
  - Example using Io.Get_Line with constant strings:

```
with Io;

procedure Get_A_Name is

begin

    Io.Put (Item => "Enter you first name: ");
    declare
        First_Name : constant String := Io.Get_Line;
    begin
        Io.Put (Item => "The name you entered is ");
        Io.Put_Line (Item => First_Name);
    end;
end Get_A_Name;
```

I/O Facilities (cont.)

- Package !Io.Window_Io
  - Is used for window-based user interfaces
  - Supports menu, form, box graphics interfaces available in Software Library Catalog
  - Provides access to raw character stream
- Package !Io.Pipe provides message passing between jobs
  - Higher performance than reading and writing files
  - Implicit queuing of messages
  - Correct synchronization properties (can be opened by two jobs simultaneously)

33 System Management: System Administration December 1992 RATIONAL

I/O Facilities (cont.)

- Packages !lo.Polymorphic_lo and !lo.Polymorphic_Sequential_lo
  - Support writing and reading of multiple, user-defined data types
  - Are used in archive databases for later use

-
  -
  - Often, a user will write a program that creates a database of information in memory. This package can be used to store that information permanently in a file for later restoration and reuse.

Tools

- Package String_Utilities augments Ada's string-handling facilities
  - Case conversion
  - Conversion between numeric values and strings
  - Substring location
- Packages Unbounded_String and Bounded_String provide dynamic-length string handling
  - Conversion between strings and variable strings
  - Copy, move, append
  - Insert, delete, and replace of characters or substrings

35 System Management: System Administration December 1992 RATIONAL

Tools (cont.)

- Generic packages List, Set, Map, Queue, and Stack provide abstract type operations
- Package Table_Formatter displays a formatted table of data
- Package Table_Sort_Generic sorts a table containing any type of data

36 System Management: System Administration December 1992 RATIONAL

Tools (cont.)

- Package Debug_Tools provides programmatic access to functions that:
  - Cause a break to be recognized by the debugger
  - Display messages in the Debugger window
  - Specify symbolic task names that can be referenced when debugging
  - Provide user-defined display of object values
  - Get the name of any raised exception

37 System Management: System Administration December 1992 RATIONAL

Tools (cont.)

- Package System_Utilities provides access to system information
  - Get CPU time consumed by job
  - Get current user or session name
- Package Time_Utilities provides additional facilities for manipulating time
  - Manipulate durations
  - Convert between time formats and string representations
- Package Profile provides facilities for determining command-error response and log formats

38 System Management: System Administration December 1992 RATIONAL

Exercise: Building Simple Tools

1. Using packages System_Utilities and Time_Utilities in a Command window, write a program that measures and reports the elapsed and CPU time for the execution of a fragment of Ada code or procedure

   Test the following constructs:

   a. A delay statement for 10 seconds

   b. A loop with a large number of iterations calling a simple procedure with a null body

2. Write a login procedure that executes differently for various session logins. Discuss the approach in class first.

39 System Management: System Administration December 1992 RATIONAL

System-Programming Interfaces

- Ada interfaces to the editor, directory system, DIANA
- Package !Tools.Object_Editor
  - Provides access to Environment editor
  - Provides pathnames for selections, images, cursor position
  - Allows tools to work relative to selection, image, cursor
- Package !Implementation.Diana
  - Is used for building analysis tools
  - Provides access to semantic resolution of the compilation system

Although these packages are useful, they have higher-level counterparts that tend to be less subject to changes when the Environment is upgraded. Packages in !Tools.Lrm should be used for analyzing Ada programs, Directory_Tools should be used for working with directory information, and Activity_Tools should be used when working with activities.

-
-
  - Remember that every interface to the user is an editor, so this can be very useful.
  -
  -
- Packages in the directory !Tools.Lrm provide a higher-level interface to Diana and are the preferred way to access this information.
  -
  -

System-Programming Interfaces (cont.)

- Package !Tools.Directory_Tools
  - Provides programmatic interface to the Environment library system
  - Defines type Object.Handle similar to Text_Io.File_Type
  - Defines type Object.Iterator for lists of handles
  - Resolves pathnames to handles/iterators
  - Provides traversal to parent, enclosing, world, subunits
  - Provides size and update statistics
  - Generates Ada unit dependencies

- Well-commented specs are available.
  -
  -
  -
  - Wildcard names resolve to iterators. You can then iterate through a sequence of handles to perform an operation. Reverse: Handle to pathname function is also available.
  -
  - For example, it provides information about when and who last updated the file.
  -

Distribution of Tools

- Various methods include:
  - Copy program into "release" library
  - Add skin to "release" library
  - Keep program in place and add searchlist entry
  - Add link to the tool in a library already on the searchlist
- Use of Library.Freeze procedure prevents inadvertent changes
  - Frozen units can be viewed but not modified

-
  -
  - A skin is simply a local procedure that calls another procedure, passing each parameter to the main procedure.
  - Essentially, we want to make the program visible to users. To do this we can add the library, in which the program resides, to the searchlist.
  - Adding a link to a library already in the searchlist is usually the best approach. Ensure that the entry in the search list for this library indicates that its links are to be searched.
-
  -

Distribution of Tools (cont.)

- Use of pragma Main improves performance
  - Applies only to library subprograms
  - Saves link time when executed
  - Still requires loading
  - Main programs are demoted to installed state when any unit in their closure is demoted

- The pragma is inserted immediately after the declaration of the main unit. Main programs load all required code segments during code generation.
  - A library subprogram is a procedure or function that exists at the library level.
  -
  -
  -

Distribution of Tools (cont.)

- Create loaded main programs: Compilation.Load
  - Loaded image is stored with an object of subclass Loaded_Main
  - Similar to program objects on other systems
  - Unit remains executable even if changes to closure are made
  - Only subprogram specification is visible

- This is an optional second step.
  -
  - The VAX and MS DOS call these .exe files.
  - Loaded main programs are not demoted like simple main programs. Thus, you can reexecute these even as the program is undergoing change. Loaded_Main objects can be moved to a new context. They do not require the links that were necessary for compilation.
  - There is no corresponding body.

Exercise: Using Directory Tools

1. In a command window off your home library, enter this code:

```
declare
   use Editor, Library, Common;
   package Object renames Directory_Tools.Object;
   package Naming renames Directory_Tools.Naming;
   Iter : Object.Iterator; An_Object : Object.Handle;
begin
   Iter := Naming.Resolution ("@");
   while not Object.Done (Iter) loop
      An_Object := Object.Value (Iter);
      Io.Put_Line (Naming.Simple_Name (An_Object));
      Object.Next (Iter);
   end loop;
end;
```


45 System Management: System Administration December 1992 RATIONAL

Exercise: Using Directory Tools (cont.)

2. Enter other naming expressions for the input to Naming.Resolution
3. Try using the Unique_Full_Name function from package Naming in Directory_Tools.
4. Try functions in package Statistics.

46 System Management: System Administration December 1992 RATIONAL

System Daily Message

- The daily message is located in the text file !Machine.Editor_Data.Daily_Message
  - Is displayed with the command What.Message
  - What.Message should be in the Login procedure for each user
  - Is typically used to announce updates to the system, expected system downtime, and such

47 System Management: System Administration December 1992 RATIONAL

Sending Messages

- Messages can be sent to users with the Message.Send command
  - Messages are displayed in the message window, and should be limited to one or two lines.
  - A user must be logged in to receive the message.
  - The sender is notified if a message is not sent.
- A message can be broadcast to all logged in users with the Message.Send_All command.
  - Messages sent using Message.Send_All are also displayed on the operator's console.

48 System Management: System Administration December 1992 RATIONAL

User Defaults

- Several objects define initial conditions when users are created
  - The objects may be edited by the system manager to define systemwide policies
  - All objects are located in !Machine
- !Machine.Search_Lists.Default: Defines the initial searchlist for a user session
- !Machine.Editor_Data.@_Commands:Defines the default keymap for the session
  - Site-specific key bindings can be added

49 System Management: System Administration December 1992 RATIONAL

User Defaults (cont.)

- !Machine.@_Acl_Suffix files
  - User_Acl_Suffix: Additional ACL for user's home library
  - User_Default_Acl_Suffix: Additional default ACL for the user's home library

-
  - This is in additional to the user's group.
  -

System Access

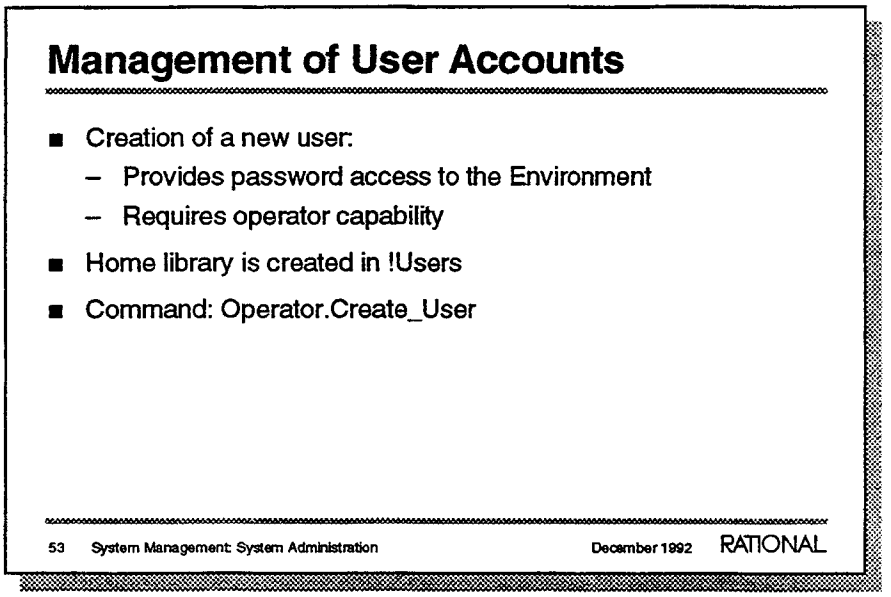
- Management of user accounts
- Access control lists – ACLs
- Operator capability
- CMVC ACLs
- Job control
- Freezing objects

51 System Management: System Administration December 1992 RATIONAL

Components of a User Account

- Username
- Password
- Home library
- One or more sessions
- Searchlist
- Session switches (optional)
- Customized login procedure

52 System Management: System Administration December 1992 RATIONAL



- A basic function of the system manager is creating and deleting user accounts.
  - A password policy can be set. This will be discussed later.
  - Only users with operator capability can perform this operation. Obtaining operator capability will be discussed later.
- All users have a home world located in !Users.
- Important Parameters for Operator.Create_user:
  - User: Specifies the name of the new user. The user's name must be a legal Ada name.
  - Password: Specifies the user's password. This can be any string. It is a good idea to embed a non-alphabetic character (#,%,*, and so on) for the initial password. As mentioned above, password policies for length and expiration will be discussed later.

  Example:

```
Operator.Create_User (User => "Bob", Password =>
   "water^ski");
```

  -- Creates user Bob with password "water^ski"

  -- Creates a !Users.Bob home library with a default session S_1

For more information on creating user accounts see Chapter 3 "Managing User Accounts," in your *System Manager's Guide.*

Management of User Accounts (cont.)

- Initialization options:
  - Create initial login procedure
  - Define initial links
  - Create initial session switches and searchlists
  - Set default activity for a session
  - Establish initial key bindings
- System manager can build a single command procedure that:
  - First creates a user
  - Then executes the required initialization operations

54 System Management: System Administration December 1992 RATIONAL

Passwords

- Passwords can contain any ASCII characters
  - Visible characters should be used so that they can be typed at login
- Changing passwords
  - Command: Operator.Change_password
- Password Policies
  - Minimum password length can be required
  - Passwords can be set to expire if not changed periodically
  - Command: Operator.Set_Password_Policy

-
-
  - Example:

```
Operator.Change_Password (User => "Bob",
          Old_Password => "water^ski",
          New_Password => "ford*explorer",
          Response => "<PROFILE>");
```

- Operator capability is required to set the password policy.
  - Example:

```
Operator.Set_Password_Policy (Minimum_Length => 0,
          Change_Warning => Operator.Days'Last,
          Change_Deadline => Operator.Days'Last);
```

  - Change_Warning is the time, after the last change when the user is notified upon login that the current password will soon expire if not changed.
  - Change_Deadline is the time, after the last change when the user will be unable to login if the password is not changed.

For more information, see Chapter 3 "Managing User Accounts," in your *System Manager's Guide.*

Removal of User Access

- Command: Operator.Delete_User
  - Prohibits future logins
  - Does not destroy the user's home library
- Users can be reestablished if necessary
  - Access control granted to the old user is *Not* transferred to the new user
  - Temporary deactivation can best be accomplished with access control or by changing the user's password

- Example:

```
Operator.Delete_User (User => "Bob",
                      Response => "<PROFILE>");
```

  Operator capability is required.

  –

  –

- The new user will *not* have access to functions which they had previously, because a new access group is created.

For more information, see Chapter 3 "Managing User Accounts," in your *System Manager's Guide.*

System User Identity

- Can be thought of as the Environment user
  - Does not appear in !Users
  - Cannot be deleted
  - Several Environment displays use the name *SYSTEM
- Performs all system initialization at boot time

- You cannot log in as 'System'. There is no super user.
  -
  -
  -
- These will be discussed later.

Exercise: Creating and Deleting Users

1. Log in as Operator with your username as your session.
2. Create a user with a name and password of your choice.
3. Visit the home library of the new user.
4. Log in as the new user to verify that access has been established

58 System Management: System Administration December 1992 RATIONAL

Access Control

- Controls access to worlds
- Controls access to individual objects
  - Ada units
  - Files (text, binary, activity, and so on)
- Controls execution of programs and commands

- Access to objects in directories is controlled by the access granted to the nearest containing world.
-
- Read access is required for execution capability.

Access Control (cont.)

- Can be used to:
  - Isolate projects on the same machine
  - Exclude unauthorized users from a project or machine
  - Prevent accidental modifications or deletions
  - Enforce design decisions
  - Limit access to certain privileged commands

- Most projects will be using CMVC_ACLs to limit access within the project space. These will be discussed later.

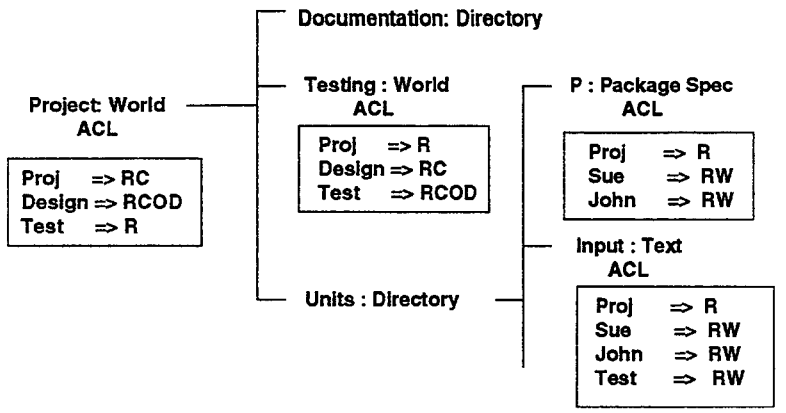
For more information, see Chapter 4 "Access Control," in your *System Manager's Guide.*

Access Lists (ACLs)

- ACLs are the basis of access control
- Each world and objects within a world have their own ACL
- ACLs consist of a list of groups and the access granted to that object

- ACLs are the underlying mechanism used for CMVC_ACLs.
-
- See the diagram on the next slide.

Access Lists (ACLs) (cont.)



Access Lists (ACLs) (cont.)

- Some objects have no ACL
  - Access to these objects is governed by the ACL of the nearest enclosing world
    - Directories
    - Session objects
- Access lists can contain a maximum of seven groups

See the diagram on the previous slide.

CMVC Access Control

- Restricts access to objects under CMVC
  - Subsystems
  - Views
  - Objects
- Includes these CMVC access control classes
  - Reader
  - Client
  - Developer
  - Owner

64 System Management: System Administration December 1992 RATIONAL

CMVC Access Control (cont.)

- Supports roles in the development process; for example:
  - Can control read and execution rights of objects
  - Can control importing of subsystem views
  - Can control check out, check in, and edit of objects
- Does not replace or circumvent basic ACL mechanisms

65 System Management: System Administration December 1992 RATIONAL

Package Cmvc_Access_Control

- To add groups:
  Cmvc_Access_Control.Add_Group
- To display current access-control information:
  Cmvc_Access_Control.Display
- To verify that ACLs are consistent with current CMVC access:
  Cmvc_Access_Control.Check
- To get a complete list of capabilities see the package

For more information, see the Project Management (PM) book of the *Rational Environment Reference Manual*.

Groups

- A group consists of a list of usernames
  - Groups cannot reference other groups
- Every user belongs to at least one group, which has the name of the user
- Having access to a world or object means:
  - The user belongs to at least one group in the ACL with the required access rights
- A job is granted access to an object if the user initiating the job is a member of a group with the required access

- Nesting of groups is not allowed.
  -
- The group with the same name as the username is created automatically when the user is created.
-
  -
- Jobs inherit the access right of the user who initiated the job.

Special Groups

- Public: All users on the local machine
- Network_Public: All users on the local machine and all users on other machines that access through the network
  - Users can be removed from Public and Network_Public
- Privileged: Users in this group can run jobs in privileged mode, which disables all access checks
- Spooler: Group allowed separate access to the print spooler
- System: Group whose initial member is the system identity (*SYSTEM)

These groups are predefined by the Environment. Do not delete these groups.

- Users are added to this group by default when the user is created.
- If the machines are not on a network, this is a nonissue. Users are added to this group by default when the user is created. This is a useful way to access objects though the network.
  –
- Privileged mode works only on a job-by-job basis. It is not possible to run an entire session in privileged mode.
-
-

Special Groups (cont.)

- Initial users/groups:
  - Operator: Users in this group have operator capability, which allows execution of system-management commands
  - Rational: User account for Rational technical representatives

-
  - The operator user is granted operator capability by default. Operator capability will be discussed in detail later.
  -

System Identity

- *SYSTEM is an implicit member of:
  - Public
  - Network_public
  - Privileged
  - System
- One of these groups must have access to objects required for machine initialization

70 System Management: System Administration December 1992 RATIONAL

Access to Worlds

- Four kinds of access:
  - (O) owner
  - (R) read
  - (C) create
  - (D) delete

Remember: directories do not have ACLs.

Access to Worlds (cont.)

- Read access
  - Can look at the world and its contents
  - Must have read access to every world in the world's full pathname
  - Must have read access to the switch file associated with the world
  - If read access is not granted, Environment acts as if it does not exist
- Create access
  - Can create new objects in the world
  - Can create new versions of existing objects in the world

- If read access is not granted for an object, it will not be found by the Definition command.
  -
  -
  - The switch file is opened when the world is displayed.
  -
- Create access is required for creating new and editing existing objects in a world. When editing an object, a new version is created.
  -
  -

Access to Worlds (cont.)

- Owner access
  - Can change the ACL for the world and for any objects in the world
  - Can change the links for the world
  - Can associate a switch file with the world
  - Can freeze/unfreeze the world or objects in the world
- Delete access
  - Can delete a world

73 System Management: System Administration December 1992 RATIONAL

Access to Nonlibrary Objects

- Two kinds of access rights: read and write
- Read access
  - Can look at an object
  - Must have read access to every world in the object's full pathname
  - If read access is not granted, the Environment acts as if the object does not exist when a read is attempted
  - If read access is not granted, the unit cannot be executed

-
- To execute an Ada unit, read access is not required for the entire closure of the object, only to the object itself.
  -
  -
  -
  -

Access to Non-Library Objects (cont.)

- Write access
  - Can make changes to the object
  - Can delete the object
  - Can demote and promote an Ada Unit

- Write access is required to demote or promote a specific object. Dependent objects can be demoted without any access.
  -
  -
  -

Default ACLs for New Objects

- Each world has a default ACL
  - Defines the initial ACL for new nonlibrary objects created within the world
- New versions of existing objects inherit the ACL of the previous version
- New worlds inherit their ACLs from the ACL of the enclosing world, and their default ACLs from the default ACL of the enclosing world

76 System Management: System Administration December 1992 RATIONAL

Default ACLs for New Objects (cont.)

- ACLs (and default ACLs) for user home libraries are created by concatenating the following:
  - Read and write access for the newly created user group
  - Machine default defined in User_Acl_Suffix and User_Default_Acl_Suffix in !Machine

77 System Management: System Administration December 1992 RATIONAL

Commands

- Display the ACL for an object: Acl.Display
- Set the ACL of an object: Acl.Set
- Set the default ACL of a world: Acl.Set_Default
- Add a group or list of groups to the ACL of an object: Acl.Add
- Create an ACL group: Operator.Create_Group
- Add a user or users to a group: Operator.Add_To_Group
- Display the users who are members of a group: Operator.Display_group
- Delete a group: Operator.Delete_Group

- Default_Display also exists. Sample output from ACL.Display:

```
-------------------------------------------------------
!USERS.MJF % ACL.DISPLAY          STARTED 11:45:51 AM
-------------------------------------------------------
!USERS.MJF.TEST'V(2)      : MJF=>RW,NETWORK_PUBLIC=>RW
```

- List of groups and their access can be set. Examples of Acl.Set:

```
Acl.Set (To_List => "Network_Public => RWCOD",
    For_Object => "!Local.Project_Status");
Acl.Set (To_List => "Testing => R, Development => RW",
    For_Object => "!Projects.Sample_Project.This_Area");
```

- Lists and groups can have their default ACLs set.
- Acl.Add_Default also exists.

```
Acl.Add (To_List => "Mjf => RW",
    For_Object => "!Projects.Sample_Project");
```

- Acl.Delete_Group exists.
- Acl.Remove from group exits.

```
Operator.Add_To_Group (User => "Mjf",
         Group => "Development",);
```

-
-

Exercise: Using Access Control

1. Traverse to your home library
2. Display the ACLs of your home library
3. Create a new group called Test_Group
4. Grant read access to your home library
5. Remove the group Test_Group
6. Log out, return to your operator session, delete the user which you created in the first exercise, and delete the home library
7. Verify that the user cannot log in after being deleted

79 System Management: System Administration December 1992 RATIONAL

Libraries That May Need Control

- !Users
  - Users may want to control access to objects in their home worlds
  - Users also may want to control their personal searchlist in !Machine.Search_Lists
- !Model
  - You can limit changes of models to the project-design staff
  - You can prohibit the accidental deletion of an important world

80 System Management: System Administration December 1992 RATIONAL

Libraries That May Need Control (cont.)

- !Local
  - This directory should contain site-developed tools and data
- !Projects
  - A world like this can be created to hold project software
- !Commands
  - You may want to remove read access to prohibit execution of certain commands
- !Machine
  - This library contains both system and user data
  - Access control should be applied with great care

-
  -
-
  -
- This should be done with caution. Limiting access to Environment commands can be dangerous. You need a good plan. Individual subprograms inside packages cannot be controlled separately.
  -
-
  -
  -

Predefined Access Controls

- Predefined access levels
  - None: All users can do anything
  - Open: Users need to change ACL to perform some function
  - Safe: System and users are protected
  - Secure: Limited network access, less read access
- Commands are located in !Commands.System_Maintenance
- Set access control to desired level: Set_Universe_Acls
  - Parameter setting produces a printable table

-
  - None (level 0) does not place access-rights restrictions on users. This is not recommended.
  - Open (level 1) allows all users to gain nonrestricted access. Objects are protected, but any user can change ACLs to gain access to all objects.
  - Safe (level 2) is designed to protect users and the system. Only a user logged in as operator can change ACLs in order to create new, nonuser libraries or to gain access to objects that are accessible to all users at levels 1 or 2.
  - Secure (level 3) is like level 2, but has more restricted network access and read access. This level is as restrictive as the system can be and still run; it will prevent most users from executing operator commands.
-
-
  -

Predefined Access Controls (cont.)

- Compare ACLs against secure settings:
  Check_Universe_Acls
  - Reports problems of more restrictive settings

-
  -

For more information, see Chapter 4 "Access Control" in your *System Manager's Guide.*

Required Access Control

- The Environment is shipped with the required access control for correct system functioning
  - Correct settings can be checked with the System_Maintenance.Check_Universal_Acls command
- Changing the following may inhibit normal system functions
  - *SYSTEM must have read access to *All* objects in !Machine
  - *SYSTEM must have complete access to the following objects in !Machine: Accounting, Devices, Groups, and Error_Logs directories
  - *SYSTEM requires access to all user home worlds in !Users

84 System Management: System Administration December 1992 RATIONAL

Required Access Control (cont.)

- All users must have complete access to the following objects in !Machine:
  - Temporary directory
  - Switch_Definitions object
  - Users and Groups directories
  - Devices directory
  - Editor_Data directory

85 System Management: System Administration December 1992 RATIONAL

Operator Capability

- Operator capability controls who can:
  - Create and delete users
  - Execute other operations in package Operator
- Users who have write access to the object !Machine.Operator_Capability have operator capability
- Write access gives members of the particular group operator capability
- Access should be limited
- *SYSTEM group must be granted operator capability
  - This may not be set by default

-
  -
  -
-
-
- Should be limited to the system-management team and other key personnel.
- *SYSTEM may need to be added to Operator_Capability.

Privileged Group

- Members can override access control with Operator.Enable_Privileges
  - This applies only to the current job
  - It is reset when the job terminates
- Membership should be controlled strictly

- Example:

```
Operator.Enable_Privileges (Enable => True);
Library.Delete (Existing => "<SELECTION>");
```

  -
  - This command should precede the command that requires the overriding of access control. If used in a command window by itself, this command has no effect.
- As they say on TV, membership has its privileges...

Exercise: Verifying Required Access

1. Verify that the *SYSTEM user has operator capability
2. Verify that the *SYSTEM user has read access to all objects in !Machine
3. Verify that the *SYSTEM user has access to all users in !Users

If running Check_Universe_Acls, ignore messages about missing objects if certain products are not installed on the machine. For example, if mail is not installed, the object !Machine.Transfer.Distribute will not be on the system.

Job Control

- System manager may need to:
  - Determine what jobs are executing
  - Disable or kill runaway jobs
  - Force logoff
- What.Jobs displays current executing jobs
  - Display is updated every 10 seconds (10 seconds is the default)
  - Use [Object] [!] to expand the display to include all jobs
- Alternate command: Show_Jobs

-
  -
  -
  - Force logoff would be used only as a last resort.
- Sample output of What.Jobs
  -
  - Only running jobs are displayed unless the image is expanded

```
  User   Session Job S Elapsed    CPU     % CPU Cache Disk Job Name
======= ======= === = ======== ======== ===== ===== ==== ========
        SYSTEM    4 R  5/09:18  3:39:17  2.86    11K 563K System
        DAEMON    5 R  5/09:18  6:49:48  0.00     71 1046 Daemons
JPF     S_1     185 I 43:41.96    49.71  1.47    144  230 [JPF.S_1 Editor]
NETWORK NETWORK 188 I 16:50:44     0.00  0.00      7    0 Archive Server
KJH     MAIL    197 I  4:13.55     9.78  0.35    252   96 [KJH.MAIL Editor]
MJF     MAIL    200 I  9:13.06    28.44  7.07    328  141 [MJF.MAIL Editor]
        SYSTEM  201 I  8:44:21     0.00  0.00      1    0 Error Log Monitor
JRG     S_1     208 I 45:41.28    40.38  0.31    146  177 [JRG.S_1 Editor]
        SYSTEM  209 I  5/08:57    11.20  0.00      5   27 *Login: 249
RN      KERMIT  211 I    54.07     1.37  0.00    307   38 [RN.KERMIT Editor]
        SYSTEM  212 I  5/08:57    10.29  0.00      5   35 *Login: 248
PSM     MAIL    214 I  5:24.45     0.86  0.00     68    0 [PSM.MAIL Command]
        SYSTEM  215 I  5/08:57     8.48  0.00      3   67 *Login: 247
JRG     S_1     216 I 45:39.26     3.47  0.00     56   17 [JRG.S_1 Command]
```

Job Control (cont.)

- Job.Disable stops temporarily the execution of a job
  - Used to stop runaway jobs
  - Job numbers can be found with the What.Jobs or What.User Commands
- Job.Enable starts execution of a disabled job
- Job.Kill stops permanently the execution of a job
  - Can kill only user-spawned jobs
  - Requires a session parameter if the job was initiated by another session

- `Job.Disable (The_Job => 178, The_Session => "!Users.Bob.S_1");`
  -
  -
- `Job.Enable (The_Job => 178, The_Session => "!Users.Bob.S_1");`
- `Job.Kill (The_Job => 178, The_Session => "!Users.Bob.S_1");`
  - Editor jobs cannot be killed this way in Environment releases before Delta 3.0 (D_12_6_5); however, editor jobs can be killed with Job.Kill in the Delta 3.0 (D_12_6_5) release of the Environment or later.
  -

Job Control (cont.)

- Operator.Force_Logoff terminates a user's session
  - Can kill sessions with wedged or lost I/O ports
  - Requires port number from What.Users display
  - Always use Commit_Buffers => True to save work

- Example:

```
Operator.Force_Logoff (Physical_Line => 240,
   Commit_Buffers => True,
   Response => "<PROFILE>");
```

Telnet

- Provides a virtual-terminal interface to other machines on the network
- Establish a session with another machine using Telnet.Connect
  - Normally only the remote-machine name is specified
  - All other defaults are derived from the user's session switches
- To break the Telnet session use Telnet.Disconnect

-
- Important parameters:

  Remote_Machine: Specifies network name of desired machine.

  Session: Specifies when creating multiple telnet sessions.

  Escape: Specifies character used to return to the original session; [Control] - [] is the default.

  Example:

```
Telnet.Connect (Remote_Machine => "Capitol",
   Session => 1,
   Escape => Telnet_Profile.Escape,
   Escape_On_Break => Telnet_Profile.Escape_On_Break,
   Terminal => System_Utilities.Terminal);
```

- Important parameters:

  Remote_Machine: Specifies machine name of the connection.

  Session: Specifies session number if multiple sessions have been created.

File Transfer

- File Transfer Protocol (FTP) provides the ability to move files from one machine to another
- Basic method:
  - Connect and log into remote machine
  - Transfer files to/from remote machine
  - Disconnect from remote machine
- Combined operations are available

- This is standard protocol supported by most networking packages, including Excelan and Wollongong. We use TCP/IP, a standard Ethernet protocol.
-
  -
  -
- For example, operations to connect, log in, move the files, and disconnect can be combined to execute as one step.

File Transfer (cont.)

- Session switches can be used to set default parameters for:
  - Remote-machine name
  - Login name and password (a remote-passwords file can be used)
  - Password prompting, if preferred over an encrypted passwords file

94 System Management: System Administration December 1992 RATIONAL

File Transfer (cont.)

- To establish an FTP connection to a remote machine, use Ftp.Connect
- To move a file from the local machine to the remote machine, use Ftp.Store
- To move a file from a remote machine to the local machine, use Ftp.Retrieve
- To log out from the remote session and break the FTP connection use Ftp.Disconnect

-
- Important parameters:
  - From_Local_File: Specifies the name of the local file.
  - To_Remote_File: Specifies the name of the new file.
- Important parameters:
  - From_Remote_File: Specifies the name of the local file.
  - To_Local_File: Specifies the name of the new file.
-

File Transfer (cont.)

- To connect, login, move a file from the local machine to the remote machine, and disconnect, use Ftp.Put
- To connect, login, move a file from a remote machine to the local machine, and disconnect, use Ftp.Get
- To do the same thing as Ftp.Get, but transfer a specified list of files, use Ftp.Get_List

Many people write skins around these to make the transfers easier. Between R1000's, these operations are easier with commands from Package Archive.

Exercise: Transferring a File

1. Traverse to your home library and create a command window
2. Enter Ftp.Get and hit [complete]
3. Enter !Users.<your username>.login'body for the From_Remote_File parameter
4. Enter Login_Body for the To_Local_File parameter
5. Enter the appropriate values for the Remote_Machine, Username, and Password parameters. (Instead of going to another machine, you are just going out to the network and back; this simulates it)

This will be a simple loopback exercise. It simulates copying a file across the network to another machine, but in actuality the user only uses one machine in this exercise.

Exercise: Transferring a File (cont.)

6. Execute the command
7. Compare Login'body to Login_Body
8. Enter other FTP commands and explore their parameters in your command window

98 System Management: System Administration December 1992 RATIONAL

Package Archive

- Archive enables saving and restoring single or multiple objects, and copying on the same or different R1000s.
- Differs from system backups because it can be used selectively to save and restore specific objects
- Four procedures:
  - Archive.Copy
  - Archive.List
  - Archive.Restore
  - Archive.Save

99 System Management: System Administration December 1992 RATIONAL

Package Archive (cont.)

- Archive.Copy:
  - Copies one or more objects from one location to another
  - R1000 to same R1000
  - R1000 to another R1000
  - Utilizes Rational Networking-TCP/IP

Example:

```
Archive.Copy (Objects =>
   "!!Bud!Local.Printing_Tools.Unix_Print",
              Use_Prefix => "*",
              For_Prefix => "*",
              Options => "promote",
              Response => "<PROFILE>");
```

Package Archive (cont.)

- Archive.Save
  - Writes one or more objects onto a tape or library, preserving hierarchical structure.
- Archive.Restore
  - Reads some or all objects from a tape or library generated by a Save, and rebuilds the original hierarchical structure from which they were saved.
- Archive.List
  - Listing of the names of the objects that were archived by the Save procedure

- Example:

```
Archive.Save (Objects => "!Projects.Ssfp",
              Options => "R1000",
              Device => "MACHINE.DEVICES.TAPE_0",
              Response => "<PROFILE>");
```

- Example:

```
Archive.Restore (Objects =>
   "!Projects.Ssfp.This_Selected_Object",
                 Use_Prefix => "*",
                 For_Prefix => "*",
                 Options => "R1000",
                 Device => "MACHINE.DEVICES.TAPE_0",
                 Response => "<PROFILE>");
```

-

For more information, see the Library Management (LM) book of the *Rational Environment Reference Manual*

Exercise: Saving/Restoring Data

1. Make an archive of the !Users.Operator library
2. Delete a file in !Users.Operator.
3. Restore the archive and verify that the file reappears.

1. The archive operation (Archive.Save) takes less time than a full backup, but requires several minutes.
2. The command is Library.Delete(filename);. If you are not sure what to delete, check with your instructor.
3. The command is Archive.Restore. If you do not specify the objects to restore, all objects on the tape will be restored.

Remote Passwords

- Are used when accessing remote hosts
- Provide a way to encrypt passwords for security purposes
- Can be set on a session-by-session basis
- Package Remote_Passwords provides a way to
  - Add entries in the remote-passwords file
  - Change entries in the remote-passwords file
  - Delete entries in the remote-passwords file

The remote-passwords file contains the following format:

host_name username password_value

-
-
- A remote-passwords file is set up on a session-by-session basis using the Profile.Remote_Passwords session switch.
-