XVM/RSX PART IV
MONITOR CONSOLE ROUTINE

CHAPTER 1

INTRODUCTION TO THE MONITOR CONSOLE ROUTINE

1.1 MCR FUNCTION TASKS

The Monitor Console Routine (MCR) functions are tasks that facilitate operator communication with RSX. These tasks consist of one permanently resident Dispatch Routine (also called the Resident MCR task) and a collection of function tasks that can be invoked to perform such procedures as entering the time and date, fixing a task in core, opening registers for debugging, and altering the relationship between physical and logical devices.

The MCR function tasks are disk-resident and typically share a core partition. These tasks can, however, be built to run in any partition in the first 32K of core. The Resident MCR task remains in core at all times and accepts user requests for specific functions as input from LUN-2. The installation manager should assign this LUN to a terminal that has been dedicated to MCR interaction (MCR terminal). The Resident MCR task then requests that a specific MCR function task be brought into the core partition from disk. This function task assumes control and performs specific operations (described in subsequent chapters of this part of the manual). Execution of MCR function tasks, like execution of user tasks, depends on partition availability and task priority.

1.2 EXECUTION OF RESIDENT MCR

The Resident MCR task must be active in order to receive requests for specific MCR function tasks. The Resident MCR task is requested initially by the terminal handler task (using the REQUEST system directive) and subsequently by MCR function tasks. The initial request is made when the user types a CRTL/C (^C) character on the terminal associated with LUN-2. When the Resident MCR task is ready to accept commands, it sends the following message to LUN-2:

```
MCR>
```

It then waits for input to be entered to the right of the prompting character (>). The operator can enter the name of an MCR function task and any parameters specific to that task.

The program that waits for operator input is the MCR Dispatcher, which issues a READ to LUN-2. From the operator input, the MCR Dispatcher

determines which MCR function task to request. Immediately after requesting the task, the MCR Dispatcher exits to allow the MCR function to execute.

Besides operator input, the MCR terminal is used as the destination of assorted system messages. Because of this, it is important that the MCR Dispatcher does not tie up the terminal with an unsatisfied READ request. If operator input has not been received within 30 seconds of the last character or the issuing of the READ request, the MCR Dispatcher issues a carriage return, cancels the READ request and exits.

The first three characters of the command input, preceded by three periods, form the name of the MCR function task requested by the Resident MCR. For example, a specification of:

```
MCR>DEVICES AND ASSIGNMENTS
```

results in a request for an MCR function task named ...DEV at task-building time. (MCR function tasks can be constructed by the user and assigned internal names in accordance with MCR naming conventions.)

1.3 MCR/SYSTEM COMMUNICATION

Two subroutines with entry points in the System Communication (SCOM) area of the Executive are used by both the Resident MCR and by MCR function tasks. The routines are FAC and IFAC. FAC is used to "fetch a character" from a line of command input. IFAC is used to initialize the FAC subroutine by reading a line of command input and setting appropriate pointers. These subroutines are invoked after the Resident MCR has been readied for input.

The internal name of an MCR function task is formed by reading a line of command input (IFAC), fetching the first three characters (FAC) and preceding these characters with three periods. Therefore, only the first three characters of an MCR function name need to be entered as input. After forming the MCR function task name, the Resident MCR task continues to fetch characters until either a space, comma, carriage return or altmode is found. After a specific MCR function task has been requested, the Resident MCR task exits. If additional control information is contained in the first line of input, it is read by the requested function task via the FAC subroutine. If additional lines of input are required by the function task, they are read using both the IFAC and FAC subroutines.

The SCOM area contains an MCR REQUEST inhibit flag (MCRRI) that is examined and set by both the terminal handler and individual MCR function tasks. MCRRI is usually cleared by an MCR function task just before the task exists. If MCRRI is zero and a CTRL/C is typed, the terminal handler requests ...MCR and sets MCRRI to 1. If MCRRI is not zero and CTRL/C is typed, the terminal handler sets MCRRI to -1. Any MCR function task that is running, periodically examines MCRRI to determine whether the operator has requested attention. If MCRII=-1, the MCR function task terminates immediately. CTRL/C can, therefore, be used to terminate an MCR function task that produces lengthy output.

1.4 MCR COMMAND CONVENTIONS

Requests for individual MCR function tasks are entered according to the following conventions:

1. Command strings are terminated by either a carriage return or an altmode. If a carriage return is typed, the Resident MCR is requested when the current MCR function task has completed execution. If an altmode is typed, the Resident MCR is not automatically requested; a CTRL/C is necessary to reestablish MCR interaction. If an error in the MCR command line occurs, the Resident MCR is requested.

   MCR functions interact with the user via LUN-2. MCR error messages are produced on LUN-3. At most installations, LUN-2 and LUN-3 are associated with the same physical device: a terminal.

2. Each element of a command string must be separated by either a comma or a space. The two separators are usually interchangeable.

3. Any number of characters (except a comma or space) can be inserted between an MCR function task name and its arguments or command-string terminator (carriage return or altmode). For example, the following forms are equivalent:

   ```
   MCR>INS TASK1

   MCR>INSTALL TASK1
   ```

   This facility is useful for improving the readability of terminal output.

4. If an error in the command string is discovered before the terminator has been typed, the line can be deleted as far back as the prompting character (>) by typing CTRL/U (^U). An "at" (@) symbol is echoed, informing the user that the command string can now be retyped. The rubout character, echoed as a backslash (\), can be used to delete the last character typed. Type one rubout for each character to be deleted.

5. If the user types LOG as the first three characters of the command string, the line is treated as a comment; not as a request for an MCR function task. Following is an example:

   ```
   MCR>LOG   THIS TASK REQUESTS
   MCR>LOG   EXECUTION OF SCAN
   MCR>REQ SCAN 50
   MCR>LOG   AT PRIORITY 50
   ```

CHAPTER 2

MCR FUNCTION TASKS

This chapter describes all available MCR Function Tasks. In all models and examples included in the following sections, certain conventions apply:

1. A space in the text indicates an actual space in the command line.

2. ▽ indicates a terminator. This may be either a carriage return or an ALTMODE; the previous section discusses the functional differences between the two.

3. Square brackets ([]) indicate optional characters.

4. Braces ({}) surrounding a stack of items indicate that one item in the stack must be selected.

5. Dots enclosed in square brackets ([...]) indicate that all previous parameters may be repeated; if a comma precedes these dots ([,...]), only the last parameter may be repeated, and repetitions of this parameter must be separated by commas. Thus by using the following form:

   ```
   REA[SSIGN] LUN[,...] new old [/...]▽
   ```

   more than one LUN can be REASSIGNed as follows:

   ```
   MCR>REA 6,8,9 TT1 TT0
   ```

   and the entire reassignment clause can be repeated as well:

   ```
   MCR>REA 6,8,9 TT1 TT0/5 DT7 DT5
   ```

6. In "Form" models, upper case characters (except LUN) indicate those required by the system, while lower case characters indicate entries which are to be specified by the user and are dependent on the particular Task being invoked.

ABO

2.1 ABORT: ABORTING TASK EXECUTION

The ABORT MCR Function task aborts the operation of a task running in either exec mode or user mode. It forces a task of lower priority than its own to EXIT.

Form:	ABO[RT] name ∇
Where:	name of task ABORTed is a string of one to six characters
Example:	MCR>ABO SCAN MCR>

After ABORT has been issued, the Active Task List is scanned for the referenced task. If found, the resumption address in the task ATL node is set to 000300, forcing the task to exit as soon as it resumes control. To force the task to regain control, its status is set to 4 so that it is ready to restart. Once the task actually exits, the EXIT directive determines whether or not I/O Rundown must be invoked for this particular task.

I/O rundown is the delaying of the availability of a core partition until all transfers to and from that partition have been stopped or have been allowed to complete.

If the ABORTed task has been activated with periodic rescheduling, the rescheduling is not cancelled. If the task is fixed in core, it remains in its partition. If the task named in an ABORT command cannot be found in the Active Task List, the System Task List is scanned. If the task is found in the STL, the following message is printed:

```
MCR>ABO SCAX
ABO-TASK INACTIVE
MCR>
```

If the task is not found in the STL, ABORT prints:

```
MCR>ABORT SCAP
ABO-TASK NOT IN SYSTEM
MCR>
```

When the ABORT MCR Function task (...ABO) is installed in the system, it is typically assigned a priority of 1. Because it scans the Active Task List starting with the node following its own, it will not abort I/O handlers or other tasks that also run at a priority of 1. Furthermore, ABORT will not allow the MULTIACCESS Monitor (TDV...) or the batch handler (BATCH) to be aborted. If an attempt is made to abort either of these tasks, ABORT prints:

```
ABO - SYSTEM TASK
```

Page Missing From Original
Document

Page Missing From Original
Document

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

ADV

2.5 ADV: ADDING A NEW DEVICE

The ADV MCR Function Task is used to add a new device to the Physical Device List (PDVL). This causes an entry for the new physical device to be constructed and placed in the PDVL. This Task has the following form:

Form:	ADV name[n] ▽
Where:	name is a legal device name (see the table below) n is an octal unit number
Examples:	MCR>ADV RF MCR>ADV DT4 MCR>

The name identifying the device to be added must be a name from the table included below, and the unit number must be valid for the device.

Table 2-1
RSX Devices

Device Name	Device
TTn	Terminal
DTn	DECtape
MTn	Magtape
RF	Fixed-Head Disk
RPn	Disk Pack
RKn	Disk Cartridge
PR	Paper Tape Reader
PP	Paper Tape Punch
CD	Card Reader
CP	Card Punch
LP	Line Printer
AD	Analog-to-Digital Converter
AF	Automatic Flying Capacitor Scanner
UD	Universal Digital Controller
CC	System COMMON Communicator
VTn	Display
VWn	Writing Tablet
XY	XY Plotter

Unit number n is only valid for device names TTn, DTn, DPn, MTn, VTn, or VWn. In addition to checking that a unit number is associated with one of these devices, the system will also verify that n is one greater than the last unit number entered for the specified device in the PDVL. Therefore if DECtape units 1 and 2 have been assigned and the following MCR Function Task is specified:

```
MCR>ADV DT4
```

An error message of the following form will be displayed:

```
UNIT NUMBER IS OUT OF ORDER
```

If all units of the specified device have already been assigned, the following will be displayed:

```
UNITS OF THIS DEVICE ARE ASSIGNED
```

If the device name specified in the command is found in the Physical Device List, the following occurs:

```
MCR>ADV LP
DEVICE EXISTS ALREADY
MCR>
```

The user should also ensure that a LUN is not associated with any of the units of the device being assigned. If no LUNs are assigned, the Handler for the device will be requested by the REASSIGN MCR Function Task before sending any I/O to the unit. This will ensure that the new PDVL entry is initialized.

ASP

2.6 ASP: ASSIGNING A PARTITION

The ASP MCR Function Task is used to change the partition assignment and, optionally, the priority of a Task built in USER mode. Changing the partition assignment causes a new pointer to the Partition Block Description List (PBDL) to be inserted in the System Task List (STL) node for the Task identified in the command. The Task named cannot be active at the time the MCR Function Task is requested.

The priority which has been specified during Task Building or installation can be overridden by ASP. If a new priority is not included in the command line, the Task runs at the previously specified default priority.

Form:	ASP partition←name [p] [,...] ▽
Where:	partition is a string of one to six characters name of Task is a string of one to six characters p is an integer representing priority in range 1-512 decimal
Examples:	MCR>ASP P.1←XYZ 500 MCR>ASP P.3←PRGM1, P.4←PRGM2 MCR>

After verifying that both the partition and Task described in the command format are valid, ASP disables the Task named, so that it will not be requested while its STL node is being modified. After modification, the Task is enabled once again.

The following examples illustrate possible forms of the command and several error messages:

```
MCR>ASP SCAX
ASP-FORMAT ERROR
MCR>ASP P.11←SCAN
ASP-CANNOT FIND PBDL NODE
MCR>ASP P.1←SCAX
ASP-CANNOT FIND STL NODE
MCR>ASP P.1←SCAN
ASP-TASK IS ACTIVE
MCR>ASP P.1←SCAN1
MCR>
```

CAN

2.7 CANCEL: CANCELLING REQUESTS FOR A TASK

The CANCEL MCR Function Task is used to CANCEL all scheduled requests for activation of a particular Task by nullifying those requests in the Clock Queue. CANCELlation does not affect current execution of the relevant Task, nor does it affect future scheduling of that Task, as DISABLE does. Note that Task CANCELlation does not remove any nodes from the Clock Queue. Nodes are removed only when the time given in the node is reached.

Form:	CAN[CEL] name ▽
Where:	name of Task CANCELled is a string of one to six characters
Example:	MCR>CANCEL SCAN MCR>

If the requested Task cannot be found, the following error message will be displayed:

```
MCR>CAN SCAX
CAN-TASK NOT IN SYSTEM
MCR>
```

COM

2.8 COMMON BLOCK: PRINTING COMMON BLOCK DEFINITIONS

The COMMON BLOCK MCR Function Task is used to construct a description of each System COMMON block currently defined. The COMMON block list is output to the device associated with LUN-3. Each description consists of the following (printed left to right, one line per COMMON block):

- COMMON block name
- COMMON block base address (octal)
- COMMON block size (octal)

To terminate output prematurely, type CTRL/C. The following is only an example; COMMON blocks described are not necessarily recommended for RSX use.

Form:	COM[MON BLOCKS]▽
Example:	MCR>COMMON BLOCKS ABC 070000 000400 FLAG 070400 001400 MCR>

If there are no COMMON blocks currently defined in the RSX system, the following message will be displayed:

```
MCR>COM
COM -- NO SYSTEM COMMON BLOCK DEFINITIONS
MCR>
```

2.9 CONSTRUCT: INSTALLING A TASK ON A USER DISK

The CONSTRUCT MCR Function task adds a task previously built by the Task Builder to the RSX system. When CONSTRUCT is invoked, the specified file, with extension TSK, is read from LUN-5. Unlike the INSTALL function, CONSTRUCT is normally used to store the core image of the task in a created file rather than in allocated space on the system disk. A LUN is included in the CONSTRUCT command line to specify the particular disk on which the created file is to reside. The file occupies contiguous blocks of disk space. Whereas INSTALL causes a node for the task named in the command to be inserted into the System Task List, CONSTRUCT does not affect the STL in any way. An entry for the CONSTRUCTed task is not inserted in this list until the EXECUTE MCR Function task actually activates the task.

Form:	CON[STRUCT] name [LUN]∇
Where:	name of task to be CONSTRUCTed is a string of one to six characters LUN is an integer representing a logical unit number currently associated with a disk (if not specified, LUN defaults to decimal 14)
Example:	MCR>CON SCAN 16 MCR>

CONSTRUCT has been implemented to allow users with a large number of tasks to store some of the more infrequently used tasks in core-image form on a user disk. The EXECUTE MCR Function task can then be used to run CONSTRUCTed tasks one or more times during each day without occupying space on the system disk for long periods of time.

DAT

2.10 DATE: RETRIEVING TIME AND DATE

The DATE MCR Function Task is used to retrieve both the date and time of day. The user types DAT or DATE, and the system outputs date and time to the device associated with LUN-3.

Form:	DAT[E] ▽
Example:	Date: November 24, 1975 Time: 30 seconds past 3:45 p.m. MCR>DATE 11/24/75 15:45:30 MCR>

DEQ

2.11 DEQ: LISTING SYSTEM DEQUES

The DEQ MCR Function Task is used to list the contents of nodes in various system lists or deques (i.e., linked double-ended queues) by entering a command in the following form:

Form:	DEQ name▽
Where:	name is list symbol A, S, B, P, D, or C
Examples:	For a listing of the PBDL: MCR>DEQ P MCR> For a listing of the STL: MCR>DEQ S MCR>

List symbols identify system lists in the following way:

Table 2-2
DEQ List Symbols

Symbol	List
A	Active Task List (ATL)
S	System Task List (STL)
B	System COMMON Blocks Description List (SCDL)
P	Partition Block Description List (PBDL)
D	Physical Device List (PDVL)
C	Clock Queue (CKQ)

If the requested list is empty, the following message will be displayed:

```
MCR>DEQ B
*** NO NODES IN LIST ***
MCR>
```

If there are nodes in the deque, the requested listing will appear on the device assigned to LUN-16, normally the line printer. An example of such a listing is included below. The following table summarizes abbreviations which may appear in the listing.

Table 2-3
DEQ Listing Abbreviations

Abbreviation	Meaning
AC	AC
ACT	Task active
AFLG	attach flag
AIR	autoincrement registers
BANK	BANK mode
BCAL	bad CAL
BP	backward pointer
BUFP	buffer pointer
CBBA	COMMON block base address
CBS	COMMON block size
CLK1	XM clock overflow count
CLK2	XM clock ticks beyond overflow count
CPT	count of pending transfers
DA+U	disk address and unit
DBA	DBA
DEVU	device unit
DISA	Task disabled
ENTR	Task entry point
EPA	EPA buffer
EVAD	address of Event Variable
EXEC	EXEC mode
FIX	Task is fixed in core
FFP	floating-point processor required
FMA1	FMA1
FMA2	FMA2

(Continued on next page)

Table 2-3 (Cont.)
DEQ Listing Abbreviations

Abbreviation	Meaning
FMQ1	FMQ1
FMQ2	FMQ2
FP	forward pointer
HWOF	handler with open files
ICL	interrupt connect location
IOT	input/output instruction permission
JEA	JEA and guard bit buffer
JMS*	JMS*
L20	L20
LR	LR
LS	link set
MAP	LUN mapping specified
MARK	mark-time request
MM	MM register buffer
MPV	memory-protection violation
MQ	MQ
NEWV	nonexistent memory reference
NOS	no shared accesses allowed
NULL	cancel or unmark
NZT	nonzero transfer
PBA	partition block address
PC	PC
POCC	partition occupied
PRIO	priority of task
PSIZ	partition size

(Continued on next page)

Table 2-3 (Cont.)
DEQ Listing Abbreviations

Abbreviation	Meaning
R1-R6	system registers 1 to 6
RADD	start or resumption address
REA	reassign inhibit
REMV	remove on exit
RO	read-only shared accesses allowed
RSEC	reschedule interval in seconds
RSIZ	resident size
RSRA	Register Save Routine entry
RTIK	reschedule interval in ticks
RW	read and write shared accesses allowed
SC	step counter
SHRPT	pointer to shared area TCNT word
SSEC	schedule interval in seconds
STAT	status of task
STIK	schedule interval in ticks
STLA	STL node address
TCNT	task use count
TEVA	trigger event variable address
TSIZ	task size
TSNR	task scheduling, no rescheduling
TSPR	task scheduling, periodic rescheduling
UNUS	unused
USR	MULTIACCESS user number
VPSIZ	virtual partition size
XR	index register
XVM	wide indirect addressing

If the user types the following command:

```
MCR>DEQ A
```

a listing in the form shown below will be produced on the line printer:

```
ACTIVE TASK LIST


011156     FP
000244     BP
042313     DSK@@@
000000
000001     PRIO
007341     PBA
000000     STLA
000003     STAT
406616     RADD LS   EXEC
007307     EVAD


014520     FP
007343     BP
042301     DSA@@@
000000
000001     PRIO
011104     PBA
000000     STLA
000003     STAT
010021     RADD EXEC
011062     EVAD


025434     FP
011156     BP
242431     TTy@@@
000000
000001     PRIO
014446     PBA
000000     STLA
000003     STAT
012275     RADD EXEC
014545     EVAD


011737     FP
014520     BP
142056     LP....
565656
000001     PRIO
023576     PBA
025124     STLA
000003     STAT
030502     RADD EXEC
030602     EVAD


025446     FP
025434     BP
042456     DT....
565656
000001     PRIO
023650     PBA
025232     STLA
000003     STAT
432006     RADD LS   EXEC
034324     EVAD
```

```
025662    FP
011737    BP
222056    RP....
565656
000001    PRIO
023452    PBA
025162    STLA
000003    STAT
420555    RADD LS    EXEC
021475    EVAD


005666    FP
025446    BP
565656    ...DEQ
040521
000002    PRIO
023400    PBA
025422    STLA
000005    STAT
015654    RADD EXEC
015553    EVAD


000244    FP
025662    BP
111722    IORD@@
040000
000010    PRIO
005664    PBA
000000    STLA
000003    STAT
005423    RADD EXEC
005635    EVAD
```

DEV

2.12 DEVICES AND ASSIGNMENTS: PRINTING A CURRENT LIST

The DEVICES AND ASSIGNMENTS MCR Function Task is used to construct a list of physical device units and the Logical Unit Numbers assigned to each. The list is output to the device associated with LUN-3. Each entry in the list consists of the following (printed left to right, one line per device):

- Physical device unit (see the table included below for legal device names in RSX)
- Logical Unit Numbers (decimal)

To terminate output prematurely, type CTRL/C.

Form:	DEV[ICES AND ASSIGNMENTS]↵
Example:	MCR>DEV DK0 1 TT0 2,3,4,12,13,14,16,20,21 TT1 TT2 TT3 TT4 TT5 RF0 LP0 DT0 DT1 DT2 MCR>

Note that unassigned LUNs (i.e., those corresponding to NONE in an MCR REASSIGN command) are not listed. The preceding is only an example; the appendix on recommended LUN assignments should be consulted for detailed assignment information.

The following table summarizes all physical devices available in the RSX system.

Table 2-4
RSX Devices

Device Name	Device
TTn	Terminal
DTn	DECtape
MTn	Magtape
RF	Fixed-Head Disk
RPn	Disk Pack
RKn	Disk Cartridge
PR	Paper Tape Reader
PP	Paper Tape Punch
CD	Card Reader
CP	Card Punch
LP	Line Printer
AD	Analog-to-Digital Converter
AF	Automatic Flying Capacitor Scanner
UD	Universal Digital Controller
CC	System COMMON Communicator
VTn	Display
VWn	Writing Tablet
XY	Reserved for Plotter

DIS

2.13 DISABLE: DISABLING A TASK

The DISABLE MCR Function Task is used to cause rejection of all future REQUEST, SCHEDULE, RUN, and SYNC commands for a particular Task. If a Task has been DISABLEd, it cannot respond to further commands until an ENABLE is issued. A DISABLEd Task is not automatically deleted from the system. Only the REMOVE MCR Function Task can cause deletion of a Task.

Form:	DIS[ABLE] name ∇
Where:	name of DISABLEd Task is a string of one to six characters
Example:	MCR>DISABLE SCAN MCR>

If the Task name cannot be found in the System Task List, the following occurs:

```
MCR>DIS SCAX
DIS-TASK NOT IN SYSTEM
MCR>
```

DOS

2.14 DOS: BOOTSTRAPPING THE DOS SYSTEM

The DOS MCR Function Task is used to bootstrap the DOS system from RSX. To invoke DOS, enter the MCR command according to the following form:

Form:	DOS ▽
Example:	MCR>DOS

2.15 DSM: DISMOUNTING A DISK

The DSM MCR Function Task is used to dismount a 'user' disk and thus disable all file-oriented I/O addressed to the disk until the next MNT command is encountered or until a LUN is REASSIGNed to that disk with a UFD specified. DSM has the following form:

Form:	**DSM Rnm∇**
Where:	**Rn is disk type RF,RP, or RK m is a valid disk unit number**
Example:	**MCR>DSM RK6 DISK IS READY FOR DISMOUNTING MCR>**

After the specified disk type and unit are checked for validity and for inclusion in the Physical Device List (PDVL), the entry in the DISK-UFD table for the named disk is zeroed as well as all entries in the LUN-UFD table corresponding to LUNs assigned to the disk.

Following is a commented example of the way in which MNT and DSM commands can be used:

```
MCR>LOG  PREPARE TO LOGICALLY DISMOUNT
MCR>LOG  A USER DISK.
MCR>DSM  RK6
DISK IS READY FOR DISMOUNTING.
MCR>LOG  THE USER DISK RK6 HAS BEEN
MCR>LOG  SUCCESSFULLY LOGICALLY DISMOUNTED.
MCR>LOG  AT THIS TIME RK6 CARTRIDGES
MCR>LOG  CAN BE CHANGED IF THE USER WISHES.
MCR>LOG  ASSUMING THAT RK6 IS PHYSICALLY
MCR>LOG  READY, IT CAN NOW BE LOGICALLY
MCR>LOG  MOUNTED SO FILE-ORIENTED I/O
MCR>LOG  CAN BE QUEUED.
MCR>MNT RK6<XYZ>
MCR>LOG  RK6 IS NOW READY FOR FILE-
MCR>LOG  ORIENTED I/O, THE NAME OF THE UFD THAT
MCR>LOG  WILL BE ASSOCIATED WITH RK6
MCR>LOG  IS XYZ.
```

Several possible DSM error messages are shown below:

```
MCR>DSM MT2
DSM-DEVICE IS NOT A DISK
MCR>DSM RP9
DSM-DISK HAS NO PDVL NODE
MCR>DSM RP1
DSM-UNIT DISMOUNTED ALREADY
MCR>DSM RP2
DISK IS READY FOR DISMOUNTING
MCR>
```

2.16 DTC: DEFINING TERMINAL CHARACTERISTICS

The DTC MCR Function Task is used to define the characteristics of terminals in the RSX system at startup time. It is necessary to specify characteristics at this time, because a variety of different terminals are supported by XVM/RSX, including:

- KSR33 Teletype unit, operating at a baud rate of 110
- KSR35 Teletype unit, also operating at a baud rate of 110
- LA30 DECwriter Data Terminal, operating at a baud rate of 110 or 300
- VT05 Display Terminal, operating at a baud rate of 110, 600, 1200 or 2400
- LA36 DECwriter Data Terminal, operating at a baud rate of 110 or 300. Since this device needs no filler characters following a carriage return, no baud rate need be specified to DTC for this device.
- VT50 Display Terminal operating at a baud rate of 2400, 1200, 600, or some rate less than 600.

The command format of DTC follows:

Form:	DTC TTn device [baud]V
Where:	n is a valid terminal unit number device is a valid terminal device (KSR33, KSR35, LA30, VT05, VT50) baud is a valid baud rate (110, 300, 600, 1200, 2400)
Examples:	MCR>DTC TT3 KSR33 MCR>DTC TT4 VT05 1200 MCR>DTC TT0 LA30 300 MCR>

The baud specification can be omitted for devices that have the default rate of 110 characters per minute.

ENA

2.17 ENABLE: REENABLING A TASK

The ENABLE MCR Function Task is used to reENABLE a disabled Task in the following way:

Form:	ENA[BLE] name ▽
Where:	name of ENABLEd Task is a string of one to six characters
Example:	MCR>ENABLE SCAN MCR>

The following illustrates several possible error messages:

```
MCR>ENA SCAX
ENA-TASK NOT IN SYSTEM
MCR>ENA
ENA-SYNTAX ERR
MCR>
```

ETI

2.18 ETIME: ENTERING TIME AND DATE

The ETIME MCR Function Task is used to set the system clock and calendar according to the following form:

Form:	ETI[ME] hour:minute:second [month/day/year]▽
Where:	hour is an integer in range 0-23 decimal minute is an integer in range 0-59 decimal second is an integer in range 0-59 decimal month is an integer in range 1-12 decimal day is an integer in range 1-31 decimal year is an integer in range 0-99 decimal and represents the last two digits of the calendar year
Example:	Time: 30 seconds past 3:45 p.m. Date: November 24, 1975 MCR>ETIME 15:45:30 11/24/75 MCR>

If an out-of-range integer is entered, the following message is displayed:

```
MCR>ETIME 15:95:30 11/24/75
ETI-SYNTAX ERR
MCR>
```

XQT

2.19 EXECUTE: REQUESTING TASK EXECUTION FROM A USER DISK

The EXECUTE MCR Function task allows the user to request execution of a task that is stored in core-image form on a user disk. Actual time of execution depends on partition availability and task priority. The purpose of EXECUTE is to prevent infrequently used tasks from occupying the system disk for long periods of time.

When EXECUTE is invoked, a node is inserted in the Execute deque (EXELH). A task called FININS is then called to locate the requested task, check its partition characteristics and transfer it to the system disk. FININS inserts a node for this task in the STL and sets the "remove-on-exit" bit in this node. Next, FININS requests that the specified task be executed. After execution, the EXIT Processor sets the "done" bit in this task STL node. A task called AUTORM removes tasks from the system when deque entries for these tasks have both "remove-on-exit" and "done" bits set. AUTORM is, therefore, an important tool for conserving space on the system disk. The SAVE MCR Function task automatically synchronizes execution of AUTORM, but the operator can request the task directly, if more frequent execution is required. MULTIACCESS also initializes AUTORM to process task execution under its control.

Form:	XQT name LUN [partition] [p]▽
Where:	name of task to be EXECUTEd is a string of one to six characters LUN is an integer representing the logical unit number associated with the user disk partition is the name of the partition in which the task is located p is an integer in decimal range 1 to 512
Example:	EXECUTE SCAN at priority 100; SCAN is now located in partition P.0 on the user disk associated with LUN-16: MCR>XQT SCAN 16 P.0 100 MCR>

The priority specified during task building can be overridden by including a priority in the command line. A partition specification is required only in certain cases. For tasks built in exec mode, if a partition parameter is included, it must correspond to that given at task-building time. Both the name and the base address of the specified partition must agree with the task-built partition or the EXECUTE function is accepted. For user-mode tasks, the partition specified at task-building time can be overridden by an explicit declaration, but this EXECUTE partition specification is not required. For both exec-mode and user-mode tasks, a check is performed by the system to ensure that the task image will fit into the partition.

Following are a few error messages that might be encountered:

```
MCR>XQT SCAX
EXE-SYNTAX ERR
MCR>XQT SCAX 16
EXE-TASK NOT IN SYSTEM
```

```
MCR>XQT SCAN 16
EXE-TASK DISABLED
MCR>XQT SCAN1 16 P.0
EXE-PARTITION LOST THROUGH RCF
MCR>
```

Note that if EXECUTE is invoked but FININS is not installed, the following message appears:

```
EXE-TASK NOT IN SYSTEM
```

A node, however, is entered in the Execute deque. If FININS is installed later, the node is still in the deque.

FIX

2.20 FIX: FIXING A TASK IN CORE

The FIX MCR Function Task is used to FIX an inactive Task in an available core partition. This dedicates the partition to the Task and provides for faster response to REQUEST, SCHEDULE, RUN, and SYNC commands. FIX does not cause a Task to be executed at the time the command is issued.

Form:	FIX name ▽
Where:	name of Task FIXed is a string of one to six characters
Example:	MCR>FIX SCAN MCR>

The following illustrates several possible error messages:

```
MCR>FIX SCAX
FIX-TASK NOT IN SYSTEM
MCR>FIX TSK2
FIX-TASK IS ACTIVE
MCR>FIX TSK 3
FIX-TASK ALREADY FIXED
MCR>
```

2.21 INSTALL: ADDING A TASK

The INSTALL MCR Function task adds a task to RSX that was previously built using the Task Builder. The Task Builder creates a binary file, with a TSK extension, as output. When the INSTALL MCR Function task is invoked, the TSK file is read from LUN-5 and stored in core-image form on the system disk. The existence of the task is then recorded in the System Task List.

The user can override a priority specified during task-building by indicating a priority in the INSTALL command line.

Form:	INS[TALL] name[p]▽
Where:	name of task to be INSTALLed is a string of one to six characters p is an integer in decimal range 1 to 512
Examples:	Priority has been set during task-building: MCR>INSTALL SCAN MCR> Priority redefined here as 10: MCR>INS SCAN 10 MCR>

If the task name specified in the command is found in the System Task List, the following message is printed:

```
MCR>INS SCANlO 300
INS-TASK ALREADY IN SYSTEM
MCR>
```

Chapter 3 contains a detailed list of all error messages that can be encountered while installing a task.

Each time that a task is INSTALLed, its existence is recorded in the System Task List in core and in the SAVE-REMOVE-INSTALL system file (special chain of blocks) on disk. This redundancy of data ensures the accuracy of system lists when the system is bootstrapped. Refer to paragraph 2.33 on the SAVE MCR Function task for additional information.

MNT

2.22 MNT: MOUNTING A DISK

The MNT MCR Function Task is used to enable file oriented I/O for all the LUNs assigned to the specified disk files handler. MNT enables such I/O by manipulating a pair of tables in the executive. The LUN-UFD table relates a UFD name to a LUN assigned to a disk files handler, consequently, this table has a one-to-one relationship with the LUT. The DISK-UFD table relates a default UFD name with a disk unit. The latter table is only used by REASSIGN when a disk is being assigned to a set of LUNs but where no UFD is specified in REASSIGN's command string. When a disk is mounted, MNT enters the UFD name into the entry of the DISK-UFD table corresponding to the disk named. This initializes the default UFD name for that disk unit in the event that REASSIGN must access that table. MNT also enters the UFD name into all zero entries of the LUN-UFD table corresponding to LUN's assigned to that disk. Note that only zeroed entries are modified since for some disk/LUN pairs, the LUN-UFD table entries may have already been initialized via REASSIGN. Hence, for any disk unit, as many UFDs can be simultaneously accessible as there are LUNs assigned to that disk. Note that it is illegal to MNT or DSM the system disk, so the default UFD specification for this disk always remains 'RSX', as initialized during System Configuration. Therefore, UFD specifications for the system disk can only be set or changed via REASSIGN.

The command format for MNT follows:

Form:	**MNT Rnm<uic>∇**
Where:	**Rn is disk type RF,RP, or RK** **m is a valid disk unit number** **uic is a valid three-character UIC**
Example:	**MCR>MNT RP3<ABN>** **MCR>**

The specified disk unit will be checked to ensure that it is a valid unit and has an entry in the Physical Device List (PDVL).

If the UFD is found in the MFD, the disk-UFD entry is set accordingly. If no starting block is specified for the UFD or if the UFD is not listed in the MFD, the directory will be initialized before the disk-UFD table entry is set.

The following illustrates several possible error messages:

```
MCR>MNT FR3<ABC>
MNT-DEVICE IS NOT A DISK
MCR>MNT RF9<ABC>
MNT-DISK HAS NO PDVL NODE
MCR>MNT RP1<ABC>
MNT-DISK NOT DISMOUNTED
MCR>MNT RP2<ABC>
MCR>
```

OPE

2.23 OPEN: EXAMINING CORE OR DISK REGISTERS

The OPEN REGISTER MCR Function Task is used to OPEN one or more registers for examination and/or modification. Because the user can change the contents of both core and disk locations, OPEN serves the purpose of an on-line debugging program. It is not recommended for those unfamiliar with the details of system operation.

The initial OPEN command for core locations has the following form:

Form:	OPE[N] register [offset] ▽
Where:	register is a valid core address in octal offset is the displacement in octal words from the beginning of the register
Examples:	OPEN core register 242: MCR>OPEN 242 000242/002325> Open core register using an offset: MCR>OPE 1000 44 001044/617522>

An effective register address consists of a register-offset combination.

The initial OPEN command for disk locations is entered as follows:

Form:	OPE[N] block Rnm [offset] ▽
Where:	block is the octal disk block number Rn is disk type RF, RP, or RK m is a valid disk unit number offset is the displacement in octal words from the beginning of the block
Examples:	Open word 27 of block 3456 on disk unit RK3: MCR>OPEN 3456 RK3 27 00027/012345> Open word 0 of block 1357 on disk unit RP1 MCR>OPE 1357 RP1 00000/666666>

The default offset in both versions of this command is zero. The disk version of OPEN causes disk addresses to be calculated mod 256(10). Only registers within the disk block OPENed can be examined and modified, until a new OPEN command is given.

If the register named in an OPEN is valid, the system outputs in octal the register's address and contents and follows the contents with a prompting character (>), as in the above example. If the user wants to change the contents of the register just OPENed, he types a number

representing the new contents of the register in the following form:

Form:	[sign][radix]contents
Where:	sign is + or - radix is O (octal) or D (decimal) contents is an octal or decimal number representing the new contents of the OPENed register
Example:	MCR>OPE 36400 D16 036420/000000>-D325

The sign and radix defaults are positive (+) and octal (O), respectively. If no number is specified after the prompting character, the contents of the register are unaltered. The line is terminated with one of the characters or combinations of characters described in the table below; ↑ and * represent graphic terminal characters. Do not confuse ↑ with the terminal's CTRL key, which is used for a different purpose.

Table 2-5
OPEN Line Terminators

Terminator	Effect
ALTMODE	Close current register - no further registers OPENed; MCR Function completed by requesting Resident MCR Task if first line of input terminated with a carriage return
carriage return	Close current register; OPEN next higher register
↑carriage return	Close current register; OPEN previous register
*carriage return	Close current register; OPEN register whose address is contained in low-order 15 bits of the current register

When any register relative to a previously OPENed register is OPENed, the new register may be modified and used to determine the next register to be OPENed. The following example scans the System Task List (a deque in core with listhead in registers 242 and 243) in order to find the endpoint; a node at 20020 is inserted at this point. All lines, except those terminated explicitly with ALTMODE characters, are assumed to end with carriage returns. Comments are included on the right.

```
MCR>OPEN 242                        open register 242
000242/002325>*                     close 242, open 2325
002325/005000>*                     close 2325, open 5000
005000/005600>*                     close 5000, open 5600
005600/006200>*                     close 5600, open 6200
006200/00242>20020*                 endpoint found (pointer
                                      register 242); replace
                                      contents of 6200 with
                                      20020, close 6200, open
                                      20020
020020/001734>242                   replace contents of 20020
                                      with 242 (new endpoint);
                                      close 20020, open next
                                      higher register (20021)
020021/001322>6200↑                 replace contents of 20021
                                      with 6200; close 20021,
                                      open next lower register
                                      (20020)
020020/000242>*                     check that contents of
                                      20020 have been modified;
                                      close 20020, open 242
000242/002325>                      close 242, open next higher
                                      register (243)
000243/006200>20020 [ALTMODE]       replace contents of 243
MCR>                                  with 20020; close 243,
                                      terminate sequence
```

Now the modifications made during this sequence are checked by examining the forward and then the backward linkages.

```
MCR>OPEN 242[ALTMODE]
000242/002325>*
002325/005000>*
005000/005600>*
005600/006200>*
006200/020020>*
020020/000242>*
000242/002325>
000243/020020>*
020020/000242>
020021/006200>*
006200/020020>
006201/005600>*
005600/006200>
005601/005000>*
005000/005600>
005001/002325>*
002325/005000>
002326/000242>[ALTMODE]
MCR>
```

The next example illustrates the examination of disk registers located on RK unit 3.

```
MCR>OPEN 3456 RK3 27                open block 3456 on RK3 at word 27
00027/012345>                       close 27, open next higher register (30)
00030/543210>                       close 30, open next higher register (31)
00031/000377>*                      close 31, open 377
00377/112233>↑                      close 377, open next lower register (376)
00376/000000>1234                   replace contents of 376 with 12345, close 376, open next higher register (377)
00377/112233>                       close 377, open next higher register (mod 256)
00000/012345>↑                      close 00000, open next lower register (377, mod 256)
00377/112233>*                      close 377, open 112233 (mod 256)
00233/776655>[ALTMODE]              close 233, terminate sequence
MCR>
```

Now register 1357 on RK3 is examined:

```
MCR>OPEN 1357 RK3                   open at beginning of block 1357 on RK3
00000/666666>                       close 0, open next higher register (1)
00001/222222>[ALTMODE]              close 1, terminate sequence
MCR>
```

OPR

2.24 OPERATOR: BATCH CONTROL FOR OPERATOR

The OPERATOR MCR Function task permits operator control of the XVM/RSX batching facility. OPERATOR commands are of the form:

```
MCR>OPR xx yy
```

Options xx and yy can be typed by the operator. Available options and the function of each batch command are described in Part VIII of this manual.

PAR

2.25 PARTITIONS: PRINTING PARTITION DEFINITIONS

The PARTITIONS MCR Function task constructs a description of each core partition currently defined. The partition list is output to the device associated with LUN-3. The description consists of the following (printed left to right, one line per partition):

- Partition name
- Partition base address (octal)
- Partition size (octal)

To terminate output prematurely, type CTRL/C. Following is an example (partitions described are not necessarily recommended for RSX use):

Form:	PAR[TITIONS]∇
Example:	MCR>PAR RFDISK 025000 003000 MCR 030000 003000 LP 033000 001400 TAPE 034400 003400 TDV 040000 040000 MCR>

This page intentionally left blank

This page intentionally left blank

2.26 RCF: RECONFIGURING CORE LAYOUT

The RCF MCR Function Task facilitates dynamic reconfiguration of core in the RSX system. At system startup, partitions and large and small nodes are initially specified. RCF allows the user to add or change these specifications and to create or change system COMMON blocks. Following are the core areas that can be specified using RCF:

- Partitions
- Common blocks
- Small nodes
- Large nodes

RCF is invoked through the resident MCR, and initiates an interaction with the user, as follows:

Form:	RCF ▽
Example:	Change only TDV partition: MCR>RCF TYPE UNITS "NAME(BASE,SIZE)" PARTITION >TDV(35000,25000) > SYS COM > SMALL NODE > LARGE NODE > TYPE N TO EXIT > RCF OK! MCR>

In response to the request for each category, the user may enter as many names as desired, as in the following:

```
PARTITION
>P.1(30000,10000)
>P.2(40000,3000)
>P.3(72000,1000)
>P.4(103000,35000)
>
```

Partitions are specified in form name (base,size), where base and size must be 400-word (octal) increments. Each entry is terminated by a carriage return. To indicate that entries in any one category have been completed, type only a carriage return in response to the prompting character. This will cause the next set of names to be requested.

In choosing names for partitions, common blocks, and small and large nodes, the user must ensure that each name is entered only once by the user. It is legitimate to respecify a name currently in the system (e.g., to enlarge the MCR partition size), but the name entered must be in the same category as the name already in the system. For example if MCR is the current name of a partition, MCR may be entered in the following example:

```
PARTITIONS
>P.0 (25000,10000)
>MCR (35000,3000)
>
```

but not in the next:

```
SYS COM
>MCR (27400,400)
>
```

Partitions and system common blocks to be modified must not be in use. If the user specifies an area of core which is in use, the following message will be displayed:

```
CORE IN USE
```

Because the system is not always able to determine whether or not common blocks are in use, the following message will appear each time an existing system common block is specified for modification:

```
SYS COM CHANGE
```

It is the user's responsibility to ensure that the block is not being used.

No specified block of small or large nodes may be larger than 32K. Node blocks may not extend above 32K, regardless of size. An error of this kind will cause the following message to be displayed:

```
SYS BLK >32K
```

This message may also appear if a partition extends above 128K. However, System common blocks and partitions may exceed 32K in size and extend above 32K.

If an error is encountered, only the line which caused the error will be ignored. Thus the user can reenter a specification which was initially incorrect. To terminate a reconfiguration, type carriage return in response to all RCF requests until the program finally types:

```
TYPE N TO EXIT
```

If N is entered, the current reconfiguration will be ignored and the original system will remain intact. The user should never attempt to exit from RCF using the ABORT MCR command.

The following example shows RCF requests and responses in greater detail:

MCR>RCF	invoke RCF
TYPE UNITS "NAME (BASE,SIZE)"	system request
PARTITION	
>P50(50000,10000)	define all partition names,
>P60(60000,14000)	bases, and sizes; allocate
>P74(74000, 2000)	in 400-word (octal) incre-
>P76(76000, 2000)	ments
>PIP(100000,3000)	
>TDV(103000,35000)	
>	exit from partition defi- nition by typing carriage return
SYS COM	
>SCB1(47000,1000)	define all system common blocks; allocate in 400-word (octal) increments
>	exit from system common defi- nition by typing carriage return
SMALL NODE	
>	exit from small node defini- tion by typing carriage return
LARGE NODE	
>	exit from large node definition by typing car- riage return
TYPE N TO EXIT	type carriage return to
>	reconfigure as specified
RCF OK!	system reconfigured
MCR>	system expects next MCR command

It is often more convenient to reconfigure in several small sessions, rather than one large one. For example, the interaction above will modify partition and system common block definitions, but will cause previously defined small and large nodes to be maintained. Many users reconfigure only one or two partitions at a time, checking the results each time with the PARTITIONS MCR Function Task.

Following are several of the most common error messages which might be encountered when interacting with RCF:

```
MCR>RCF
TYPE UNITS "NAME (BASE,SIZE)"
>MCR )40000,3000)
SYNTAX ERROR AT ')'
>MCR(40000,3000)
>MCR(30000,3000)
NAME USED
>TDV(57000,4333)
INVALID SIZE
>
```

2.27 RCP: RECONFIGURING PARTITIONS

The RCP MCR Function Task facilitates dynamic reconfiguration of partitions in the RSX system.

RCP syntax is identical to that described for the RCF MCR Function Task, except that only partitions can be defined, as follows:

Form:	RCP ▽
Example:	Change three partitions: MCR>RCP TYPE UNITS "NAME (BASE,SIZE)" PARTITIONS IO.2(65000,3000) >TDV(40000,25000) > TYPE N TO EXIT RCP OK! MCR>

If partition names refer to existing partitions, these partitions must not be in use, or the following message will be displayed:

CORE IN USE

Partitions are specified in form name (base, size), where base and size must be 400-word (octal) increments. Each entry is terminated by a carriage return. To indicate that all desired partition names have been supplied, type only a carriage return in response to the prompting character. This will cause the following message to be displayed:

TYPE N TO EXIT

If N is entered, the new partition configuration will be ignored, and the original set of partitions will remain intact.

No specified partition may exceed 32K or extend above 128K. If an error of this kind is encountered, the following message will appear:

SYSBLK > 32K

It is often useful to perform several small reconfigurations, checking results each time with the PARTITIONS MCR Function Task.

Following are several of the most common error messages which might be encountered:

```
MCR>RCP
TYPE UNITS "NAME (BASE,SIZE)"
>MCR )40000,3000)
SYNTAX ERROR AT ')'
>MCR(40000,3000)
>MCR(30000,3000)
NAME USED
>TDV(57000,4333)
INVALID SIZE
>
```

REA

2.28 REASSIGN: CHANGING DEVICE ASSIGNMENTS

The REASSIGN MCR Function Task is used to alter the Devices and Assignments List by deassigning a Logical Unit Number (LUN) from a physical device and REASSIGNing it to another device when desired. If the Handler has no LUN assignments, this function automatically causes the old I/O Device Handler Task to DISCONNECT & EXIT and the new Handler to be requested. As shown in the example below, more than one Logical Unit Number can be REASSIGNed in a single command.

When a unit is REASSIGNed, both the old device and the new device must be specified in the command line. Device names are associated with I/O Device Handler Tasks provided as part of RSX and have the following form:

Table 2-6
RSX Devices

Device Name	Device
TTn	Terminal
DTn	DECtape
MTn	Magtape
RF	Fixed-Head Disk
RPn	Disk Pack
RKn	Disk Cartridge
PR	Paper Tape Reader
PP	Paper Tape Punch
CD	Card Reader
CP	Card Punch
LP	Line Printer
AD	Analog-to-Digital Converter
AF	Automatic Flying Capacitor Scanner
UD	Universal Digital Controller
CC	System COMMON Communicator
VTn	Display
VWn	Writing Tablet
NONE	Pseudo-device denoting the null LUN-device assignment

If a device name has the form TTn, DTn, MTn, VTn, or VWn, n is a unit number which identifies one of multiple terminals, DECtape drives, disk packs, MAGtape drives, displays, or writing tablets. If n is omitted, the unit number is assumed to be zero. Where nothing is assigned to a LUN, use the name NONE.

Form:	REA[SSIGN] LUN[,...] new [<UFD>] old [/...]▽
Where:	LUN is an integer representing a Logical Unit Number currently in the system; new represents the new assignment of the referenced LUN (see above for legal names); old represents the current assignment of the referenced LUN; UFD (User File Directory) 3-character name, legal only if the new device is disk, to be associated with a LUN/disk pair.
Examples:	REASSIGN the existing Logical Unit Numbers: LUN-2=TT0 LUN-3=TT0 LUN-4=DT5 LUN-33=LP to the following devices: LUN-2=TT1 LUN-3=TT1 LUN-4=DT7 LUN-33=TT0 MCR>REA 2,3 TT1 TT0 MCR>REA 4 DT7 DT5 MCR>REA 33 TT0 LP MCR> These REASSIGNments could all be expressed in one command string: MCR>REA 2,3 TT1 TT0/4 DT7 DT5/33 TT0 LP MCR>

Under RSX, MCR Function Tasks typically use LUN-2 for command input and LUN-3 for command output. When REASSIGNing these LUNs, the user should reference both on the same command line, as shown above. If the two are separately REASSIGNed, the output LUN-3 should be altered first, so that the subsequent command can still be input from LUN-2. Thus the following sequence is required:

```
MCR>REA 3 TT1 TT0
MCR>REA 2 TT1 TT0
```

The next prompt:

```
MCR>
```

appears on TT1.

Interaction with the REASSIGN MCR Function Task may result in a variety of error messages or system queries. If the user REASSIGNs a device which still has I/O requests associated with it, the device is REASSIGNed, but the following message is printed on LUN-3:

```
MCR>REA 5 TT0 LP
REA-I/O REQUEST QUEUE NOT EMPTY ON LP
MCR>
```

This example applies to the line printer, but messages for other devices are produced in the same way.

If the user requests REASSIGNment of an attached device, the following system query is printed:

```
MCR>REA 8 TT8 TT9
REA-DEVICE ATTACHED, DO YOU WISH TO DETACH?
```

In response, the operator must type Y or N, signifying YES or NO, respectively. If the user types N, the device is not REASSIGNed. If the above message is produced following a command line that REASSIGNs more than one device, it is impossible to determine which device is attached. The user should, therefore, check LUN assignments by means of the DEVICES AND ASSIGNMENTS MCR Function task.

In certain cases, it may be desirable to REASSIGN an ATTACHed device. For example, if two terminals are available and only one is currently being used, the operator may want to switch messages from one to the other when the first runs low on paper, even if the first terminal is currently ATTACHed.

One REASSIGN message is particularly critical:

```
MCR>REA 30 DT1 RF
REA-FILES OPEN ON DEVICE -- DO YOU STILL WISH TO REASSIGN?
```

The user must respond with Y or N. This message may indicate that the referenced LUN is being used to access a disk file. If the response is Y, the disk files are closed. It is the user's responsibility to determine which disk LUNs are actually in use at any time. When the new device being associated with the LUNs specified is a disk, the user can include a UFD name in the command string. This UFD is typically accessed by the disk files handler whenever an I/O request is made to open or close a file on the LUNs indicated. If no UFD is named and the new device is a disk, REASSIGN assumes that the desired UFD can be obtained from the disk UFD table. Once REASSIGN has obtained the UFD name, it checks the disk MFD (master file directory) to be sure that the UFD exists. If the UFD is not initialized, REASSIGN automatically initializes it.

Example:

To REASSIGN LUN-23 from TT1 to UFD ABC on RK1:

```
MCR>REA 23 RK1 <ABC> TT1
```

REASSIGN cannot be used to assign devices to those LUNs dedicated to the MULTIACCESS Monitor and associated users. In some cases, however, REASSIGN can deassign (i.e., assign to "none") a LUN within the set of LUNs reserved for MULTIACCESS. If an attempt is made to violate this restriction, REASSIGN prints the message:

```
REA - LUN DEDICATED TO MULTIACCESS
```

REM

2.29 REMOVE: DELETING A TASK

The REMOVE MCR Function task deletes an inactive task from the system. If the user plans to alter and reINSTALL a task, he must REMOVE it first.

Form:	REM[OVE] name ∇
Where:	Name of task to be REMOVEd is a string of one to six characters
Example:	MCR>REMOVE SCAN MCR>

If the task name specified in the command line cannot be found in the System Task List, the following message is printed:

```
MCR>REM TSK2
REM-TASK NOT IN SYSTEM
MCR>
```

If the task is active at the time a REMOVE command is issued, the following message is printed:

```
MCR>REM SCAN
REM-TASK ACTIVE
MCR>
```

Each time that a task is REMOVEd, its node is deleted from the System Task List in core and this action is recorded in the SAVE-REMOVE-INSTALL system file (special chain of blocks) on disk. This process ensures the accuracy of system lists when the system is bootstraped. Refer to section 2.33 on the SAVE MCR Function task for additional information.

To minimize the size of the SAVE-REMOVE-INSTALL system file, the file is searched on every REMOVE operation for a corresponding INSTALL entry. If the INSTALL entry is found, both the REMOVE and INSTALL entries are removed from the file, because they are a matched pair.

REQ

2.30 REQUEST: REQUESTING TASK EXECUTION

The REQUEST MCR Function Task is used to REQUEST the execution of a Task at a specified software priority. Actual time of execution depends on Task priority and partition availability. The priority which has been specified during Task Building or installation can be overridden when the Task is REQUESTed. If a new priority is not included in the command line, the Task runs at the previously specified default priority.

Form:	REQ[UEST] name [p] ▽
where:	name of Task REQUESTed is a string of one to six characters p is an integer in range 1-512 decimal
Examples:	Priority has been set during Task-Building or installation: MCR>REQUEST SCAN MCR> Priority redefined here at 50: MCR>REQ SCAN 50 MCR>

If the Task name specified in the command line cannot be found in the System Task List, the following occurs:

```
MCR>REQ SCAX 75
REQ-TASK NOT IN SYSTEM
MCR>
```

The following message will be produced if the Task name is found in the Active Task List, that is, is already running, waiting to run, or in the process of being aborted.

```
MCR>REQ SCAN 80
REQ-TASK ALREADY ACTIVE
MCR>
```

RES

2.31 RESUME: RESUMING TASK EXECUTION

The RESUME MCR Function Task is used to RESUME execution of a self-suspended Task according to the following form:

Form:	RES[UME] name ▽
Where:	name of Task RESUMED is a string of one to six characters
Example:	MCR>RESUME SCAN MCR>

If the user requests resumption of a Task which has not been suspended, the following occurs:

```
MCR>RES SCAX
RES-TASK NOT SUSPENDED
MCR>
```

A Task can only suspend itself by means of a System Directive; no MCR Function Task has been implemented to allow the operator to suspend Tasks.

2.32 RUN: ACTIVATING TASK EXECUTION

The RUN MCR Function Task is used to make a Task active at some future time and to reSCHEDULE this activation at periodic intervals. Actual time of execution depends on Task priority and partition availability. This command performs the same function as SCHEDULE, except that the time of execution is expressed as time from now, not as absolute time of day.

Both the initial activation time interval and the reschedule interval are expressed in terms of ticks, seconds, minutes, or hours and may not exceed one day. Permissible ranges are illustrated in the table included below.

Table 2-7
Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	T	0T-262143T
Second	S	0S-86400S
Minute	M	0M-1440M
Hour	H	0H-24H

The initial activation time interval must be included in the command line, but the reschedule interval is optional. If no reschedule interval is specified, the Task is executed only once.

The priority which has been specified during Task Building or installation can be overridden when the Task is scheduled and rescheduled to be RUN. If a new priority is not included in the command line, the Task runs at the previously defined priority.

Form:	RUN name sisu [riru] [p] ▽
Where:	name of Task activated is a string of one to six characters si is an integer representing the number of units which must elapse before the Task is activated (range given above) su is interval symbol T, S, M, or H representing the relevant unit of time ri is an integer representing the number of units which must elapse before the Task is rescheduled (range given above) ru is interval symbol T, S, M, or H p is an integer in range 1-512 decimal
Examples:	Activate execution of SCAN, at its default priority, 30 minutes from now and reschedule it every hour thereafter: MCR>RUN SCAN 30M 1H MCR> Activate execution of SCAN at a new priority of 28 10 minutes from now and reschedule it every 32 seconds thereafter: MCR>RUN SCAN 10M 32S 28 MCR>

As the above description indicates, it is not necessary to specify a priority or a reschedule interval as part of the command line. The following examples illustrate possible forms of the command and several error messages:

```
MCR>RUN SCAN 25M
MCR>RUN SCAN 25M 512
MCR>RUN SCAN 25M 3H
MCR>RUN SCAN 25M 3H 512
MCR>RUN SCAN 25M 512 3H
RUN-SYNTAX ERR
MCR>RUN SCA 29M
RUN-TASK NOT IN SYSTEM
MCR>RUN TSK 50M
RUN-TASK DISABLED
MCR>RUN TSK1 20S
MCR>
```

2.33 SAVE: BACKING UP A SYSTEM

The SAVE MCR Function task records the core image of an RSX system in the RSXIMG overlay of the DOS-to-RSX bootstrap. This makes it possible for the RSX bootstrap to reload and perform a warm start.

SAVE provides a means of updating the system (e.g., after the partition layout has been modified) and facilitates restart at development plateaus rather than crash recoveries.

NOTE

The SAVE function executes to normal completion only when the system is quiescent (i.e., no tasks are active, no I/O is in progress and no files are open). In other words, SAVE should not be executed if there are interrupts pending in the system. To be certain that the system is quiescent, SAVE checks to be sure that the Clock Queue has no entries, no I/O nodes are queued in the PDVL and the ATL has no user tasks.

SAVE is issued in the following way:

Form:	SAV[E]∇
Example:	MCR>SAVE MCR>

SAVE automatically requests the Resident MCR, even when an altmode is used to terminate the command.

When called, SAVE:

1. Cancels the tasks POLLER, NODCNT and AUTORM.
2. Checks for system quiescence and, if it is quiescent, returns all nodes in the Clock Queue to the Pool of Empty Nodes.
3. Deletes all entries in the SAVE-REMOVE-INSTALL system file on disk (see explanation on next page).
4. Turns off the clock, PI and API, and clears all flags.
5. Saves the contents of octal location 1 in R2 and location 21 in R3 (R2 and R3 are reentrant system registers).
6. Sets a transfer vector to a warm-start entry point within ...SAV in R1 (another reentrant system register).

7. Writes locations 30 through the top of core (per SCOM) at the beginning of RSXIMG.

8. Transfers control to the warm-start entry point (R1) as though the system had been restarted with the Warm-Start Bootstrap. This means that SAVing the system does not require bootstrapping to continue execution.

The RSX Warm-Start Bootstrap restores the saved core image by:

1. Loading locations 30 through the top of core (as defined when the image was recorded).

2. Transferring control to the location indicated in the warm-start entry point within ...SAV in R1.

When control is transferred to this entry point, ...SAV then:

1. Restores locations 1 and 21.

2. Turns on PI and API, and starts the clock.

3. Forces the System Task List in the saved core image to be updated with all task installations and removals performed since the last SAVE operation. This process is accomplished by reading the SAVE-REMOVE-INSTALL system file and reconstructing the STL from the STL in the saved image and the task entries in the system file.

4. Synchronizes the tasks named POLLER, NODCNT and AUTORM on the next minute to be run as follows:

Task	Scheduled	Rescheduled
NODCNT	1M	2M
POLLER	75S	1M
AUTORM	90S	5M

5. Requests the Resident MCR.

6. Exits.

In some cases, the system date and time may need to be reset after reloading the RSX system. If DOS is bootstrapped from paper tape, the date is set, but the time is zeroed. ETI must be invoked. When bootstrapping RSX-to-DOS or DOS-to-RSX, however, both date and time are passed and need not be reentered.

Following are a few examples of SAVE error messages:

```
MCR>SAVE
TASKS ACTIVE -- SAVE ERROR
MCR>SAVE
NO SAVE AREA ON SYSTEM DISK
MCR>SAVE
FLOATING POINT UNIT NECESSARY
MCR>
```

SCH

2.34 SCHEDULE: SCHEDULING TASK EXECUTION

The SCHEDULE MCR Function Task is used to SCHEDULE the execution of a Task at some future time of day and to reSCHEDULE it at periodic intervals. Actual time of execution depends on Task priority and partition availability. The SCHEDULE time is expressed in terms of absolute time of day. The reSCHEDULE interval is expressed in terms of an interval of ticks, seconds, minutes, or hours. The interval begins at the time the Task is first scheduled to be executed and may not exceed one day. The following table illustrates permissible ranges for each interval unit.

Table 2-8
Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	T	0T-262143T
Second	S	0S-86400S
Minute	M	0M-1440M
Hour	H	0H-24H

If no interval is specified, the Task is executed only once.

The priority which has been specified during Task Building or installation can be overridden when the Task is SCHEDULEd. If a new priority is not included in the command line, the Task runs at the previously defined priority.

Form:	SCH[EDULE] name hour:minute:second [riru] [p] ▽
Where:	name of Task SCHEDULEd is a string of one to six characters hour is an integer in range 0-23 decimal minute is an integer in range 0-59 decimal second is an integer in range 0-59 decimal ri is an integer representing the number of units which must elapse before the Task is reSCHEDULEd (range given in table above) ru is interval symbol T, S, M, or H representing the relevant unit of time p is an integer in range 1-512 decimal
Examples:	SCHEDULE execution of SCAN at its default priority at 1:30 p.m. and reSCHEDULE it every 30 minutes thereafter: MCR>SCHEDULE SCAN 13:30:00 30M MCR> SCHEDULE execution of SCAN at 8:30 a.m. at a new priority of 10, and reSCHEDULE it every 2/60ths of a second (60-cycle clock) thereafter: MCR>SCH SCAN 8:30:00 2T 10 MCR>

As the above description indicates, it is not necessary to specify a priority or a reSCHEDULE interval as part of the command line. The following examples illustrate possible forms of the command and several error messages:

```
MCR>SCHEDULE SCAN 13:17:05
MCR>SCHEDULE SCAN 13:17:05 512
MCR>SCHEDULE SCAN 13:17:05 250T
MCR>SCHEDULE SCAN 13:17:05 250T 512
MCR>SCHEDULE 13:17:05 512 250T
SCH-SYNTAX ERR
MCR>SCHEDULE SCAX 13:17:05
SCH-TASK NOT IN SYSTEM
MCR>SCH SCANX 14:01:00
SCH-TASK DISABLED
MCR>
```

SHU

2.34A SHUT: SHUT DOWN MULTIACCESS

The SHUT MCR Function task prevents new users from logging into the MULTIACCESS system and automatically logs off any existing users. Each existing user is logged off as he reaches a wait condition (signalled by TDV>). Running jobs are not disturbed.

When no users are logged in (number of logged in users drops to zero), the MULTIACCESS system automatically turns itself off.

If it is desired to immediately force running jobs to stop, use the OPEN MCR Function task to set the SCOM CTRL/Y flags word (location 227) to -1 (see section 2.23).

If a batch job is running, the SHUT command is not honored. The error message:

SHU - BATCH ACTIVE

is returned to the user.

2.35 SLICE: INITIATING TIME-SLICING WITHIN A PRIORITY RANGE

The SLICE MCR Function task establishes time-slicing over a priority range in the RSX system. By specifying a time SLICE, the user can set a priority or range of priorities. Tasks running at priorities in this range run at the same effective priority. Status shifts automatically to allow all tasks some time to run. SLICE has been implemented to alleviate the problems resulting from scheduling several tasks to run concurrently. This situation can result in one task using all available processing time. SLICE parameters are specified as follows:

Form:	SLI[CE] { lowpri ticks / lowpri-highpri ticks / OFF } ∇
Where:	lowpri is a decimal number representing the lowest or only priority in the range highpri is a decimal number representing the highest priority in the range ticks is the decimal number of ticks in a scan interval (quantum)
Example:	Scan the range of priorities 100 to 150 every 50 ticks: MCR>SLICE 100-150 50 MCR> Scan tasks at priority 100 every 10 ticks: MCR>SLI 100 10 Turn off time-slicing: MCR>SLI OFF MCR>

The parameters lowpri, highpri and ticks are decimal numbers. 100 is the lowest priority allowed in the time-slicing range. Therefore, the following specification is legal:

```
MCR>SLICE 100-140 10
```

but the following is not:

```
MCR>SLICE 50-200 10
```

A FORMAT ERROR message is printed if either of the following occurs:

- High priority lower than low priority
- Ticks not in the range 1 to 999

An error message is also printed if a SLICE priority is specified and a previous SLICE has not been turned off with a SLICE OFF command. For example, the following causes an error:

This page intentionally left blank

```
MCR>SLICE 200-300 2
MCR>SLICE 300-450 5
TIME SLICING IS ALREADY ON
MCR>
```

The following is valid:

```
MCR>SLICE 200-300 2
MCR>SLICE OFF
MCR>SLICE 300-450 5
MCR>
```

A few additional error messages are:

```
MCR>SLI 110-500
TICKS MUST BE NON-ZERO
MCR>SLI 2000-3000 20
ILLEGAL PRIORITY
MCR>
```

STR

2.36 STROBE: STROBE AND LIST SYSTEM DEQUES

The STROBE MCR Function Task is used to list the contents of nodes in system lists or deques (i.e., doubly linked queues). This MCR function differs from the DEQ MCR Function Task in two major ways. First, STROBE scans the entire deque and places its contents into a buffer before printing the contents of nodes in deque. This procedure is done with interrupts disabled so the deque cannot change during the initial part of this task's execution. Note that interrupts are likely to be disabled for more than 100 microseconds so users should take care not to request STROBE at times when the worst case interrupt response time is critical. Second, STROBE does not indicate the significance of the contents of the nodes except for words 2 and 3, which usually contain a name in SIXBIT form. STROBE also differs from DEQ in the format of the printout, the command string, and STROBE's ability to list arbitrary deques defined by an octal listhead location.

The command line for STROBE is defined as follows:

Form:	STR[OBE] name ∇
Where:	name is the name of a system list: STL, SCDL, ATL, CKQ, PDVL, PBDL, or an octal listhead address
Examples:	For a listing of the PBDL MCR>STROBE PBDL MCR> For a listing of some arbitrary deque (in this case the ATL) MCR>STROBE 244 MCR>

STROBE will produce its output on LUN-16 normally assigned to the line printer with the exception of error messages. If a deque is explicitly named, the name of the deque will be printed preceding the contents of the nodes. The time of day (hours/minutes/seconds/) at which the list was STROBED will also precede the contents of the nodes even if no deque was explicitly named. If the deque is empty, the message "LIST IS EMPTY" will be printed on LUN-16 following the time. On the line before the printout of each node, the name defined in SIXBIT notation by words 2 and 3 will be printed. Following the name, the contents of the node will be printed. An example of such a listing for the Active Task List follows:

Example:

```
ACTIVE TASK LIST

00/18/26/

DSK@@
011144 000244 042313 000000 000001 007355 000000 000003 406601 007323

DSA@@
014273 007357 042301 000000 000001 011072 000000 000003 010021 011052

TTY@@
026440 011144 242431 000000 000001 014221 000000 000003 012004 014320
```

```
RP....
026642 014273 222056 565656 000001 024052 025536 000003 020555 021500

LP....
025276 026440 142056 565656 000001 024176 025372 000003 030230 030600

...STR
005651 026642 565656 232422 000002 024000 026666 000005 015000 024021

IORD@@
000244 025276 111722 040000 00010 005647 000000 000003 005406 005620
```

Octal listhead addresses for all system lists can be found in the part of this manual entitled "System Organization and Lists".

SYN

2.37 SYNC: SYNCHRONIZING TASK EXECUTION

The SYNC MCR Function Task is used to make a Task active at some future time and to reschedule this activation at periodic intervals. Actual time of execution depends on Task priority and partition availability. This command performs the same function as SCHEDULE and RUN, except that the SYNChronized time of execution is expressed as a stated interval of time after the next tick, second, minute, or hour mark, not as absolute time of day.

A SYNChronization unit specifies after which unit of time the Task is activated. Thus in the following example, a Task named TASK1 is scheduled to be run 5 minutes after the occurrence of the next hour mark. If it is now 10:58:35, TASK1 will be SYNChronized at 11:05:00. The length of the interval can therefore vary depending on the time of day.

```
MCR>SYNC TASK1 H 5M
MCR>
```

The SYNC Function is useful in SYNChronizing the execution of a sequence of Tasks, one following the other.

Both the initial activation time interval and the reschedule interval are expressed in terms of ticks, seconds, minutes, or hours, and may not exceed one day. Permissible ranges are illustrated below:

Table 2-9
Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	T	0T-262143T
Second	S	0S-86400S
Minute	M	0M-1440M
Hour	H	0H-24H

The initial activation time interval must be included in the command line, but the reschedule interval is optional. If no reschedule interval is specified, the Task is executed only once.

The priority which has been specified during Task Building or installation can be overridden when the Task is SYNChronized. If a new priority is not included in the command line, the Task runs at the previously defined priority.

Form:	SYN[C] name sz sisu [riru] [p] ▽
Where:	name of Task activated is a string of one to six characters sz is interval symbol T, S, M, or H representing the relevant SYNChronization unit si is an integer representing the number of units which must elapse before the Task is rescheduled (range given above) su is interval symbol T, S, M, or H ri is an integer representing the number of units which must elapse before the Task is rescheduled (range given above) ru is T, S, M, or H representing the relevant unit of time p is an integer in range 1-512 decimal
Examples:	SYNChronize execution of SCAN, at its default priority, three minutes after the next hour mark (i.e., it is now 09:51:07; run Task at 10:03:00) and reschedule it every hour thereafter: MCR>SYN SCAN H 3M 1H MCR> SYNChronize execution of SCAN at a new priority of 21 10 seconds after the next minute mark, and reschedule it every hour thereafter: MCR>SYN SCAN M 10S 1H 21 MCR>

As the above description indicates, it is not necessary to specify a priority or a reschedule interval as part of the command line. The following examples illustrate possible forms of the command and several error messages.

```
MCR>SYNC SCAN H 30M
MCR>SYNC SCAN H 30M 512
MCR>SYNC SCAN H 30M 2H
MCR>SYNC SCAN H 30M 2H 512
MCR>SYNC SCAN H 30M 512 2H
SYN-SYNTAX ERR
MCR>SYNC SCAN H
SYN-SYNTAX ERR
MCR>SYNC SCAX H 0S
SYN-TASK NOT IN SYSTEM
MCR>SYN SCANX S 2S
SYN-TASK DISABLED
MCR>SYN SCAN1 M 2S 200
MCR>
```

TAS

2.38 TASK LIST: PRINTING TASK DESCRIPTIONS

The TASK LIST MCR Function Task is used to construct a description of each Task currently installed in the system. The Task List is output to the device associated with LUN-3. The description consists of the following (printed left to right, one line per Task):

- Task name
- Partition name
- Task default priority (decimal)
- Starting disk block (octal)
- Size of resident Task image (octal)

To terminate output prematurely, type a CTRL/C character. The following is an example of output from a TASK LIST request.

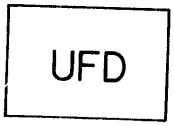
Form:	TAS[K LIST]▽
Example:	MCR>TAS TNTERM MCR 025 002510 00460 RF.... RFDISK 001 002466 01537 ...PAR MCR 002 002464 00214 ...TAS MCR 002 002461 00420 ...DEV MCR 002 002456 00501 ...RCF MCR 002 002447 02755 ...REA MCR 002 002442 01501 ...DOS MCR 002 002437 00761 ...INS MCR 002 002431 02323 ...SAV MCR 002 002424 01671 MCR>

TIM

2.39 TIME: RETRIEVING THE TIME

The TIME MCR Function Task is used to retrieve the time of day. The user types TIM or TIME, and the system outputs the time to the device associated with LUN-3.

Form:	TIM[E] ▽
Example:	Time: 30 seconds past 3:45 p.m. MCR>TIME 15:45:30 MCR>



2.40 UFD: PRINTING LUN-UFD RELATIONSHIPS

The UFD MCR Function Task is used to determine both the default UFD's and the active UFD's for all disks known to be present on the system. The following is an example of output from a UFD list request. This printout will always be written on LUN-3.

Form: UFD ▽

Example:

```
MCR>UFD
RK0-DEFAULT UFD:     NONE
RK1-DEFAULT UFD:     NONE
RP0-DEFAULT UFD:     RSX
  RSX:005,010,011,015,017,018
RP1-DEFAULT UFD:     XYZ
  ABC:040
```

In the example, RK0 and RK1 are dismounted. Hence their default UFD specification is listed as 'NONE'. For RP0, the default UFD is RSX and that UFD is associated with RP0 for LUNS 5, 10, 11, 15, 17, and 18. XYZ is the default UFD for RP1 but the only active UFD for RP1 is ABC, associated with LUN-40.

UNF

2.41 UNFIX: FREEING A CORE PARTITION

The UNFIX MCR Function Task is used to nullify a FIX function and free the core partition for use by other Tasks. If a fixed Task is active when an UNFIX is issued, the partition is made available as soon as the active Task exits. An UNFIX does not cause a running Task to be terminated.

Form:	UNF[IX] name ▽
Where:	name of Task UNFIXed is a string of one to six characters
Example:	MCR>UNFIX SCAN MCR>

The following illustrates several possible error messages:

```
MCR>UNF SCAX
UNF-TASK NOT IN SYSTEM
MCR>UNF TSK5
UNF-TASK NOT FIXED
MCR>
```

CHAPTER 3

MCR ERROR MESSAGES

This chapter describes possible meanings of error messages which may be produced during communication with any of the MCR Function Tasks described in Chapter 2.

Table 3-1
MCR Error Messages

Task	Error Message	Possible Meaning
ETI	ETI-SYNTAX ERR	Noninteger or out-of-range time or date
COM	COM - NO SYSTEM COMMON BLOCK DEFINITIONS	No system COMMON blocks currently defined in RSX system
INS	SYNTAX ERROR	Task name omitted or priority invalid
	TASK ALREADY IN SYSTEM	Task to be installed has node in STL
	PARTITION NOT IN SYSTEM	Partition name specified at, Task-Building time not available
	TASK WOULD OVERFLOW PARTITION	Task to be installed too large for available partition
	OUT OF POOL	No nodes left in pool to create new STL entry
	OUT OF DISK	No room left on the disk
	INPUT CHECKSUM ERR	Error while performing checksum processing
	INPUT PAR ERR	Error while performing parity checking
	SYS COM BLK ERR	INSTALL needs system COMMON block not currently in system

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	READ ERR ON LUN 5	Error in reading binary file from LUN-5
	DISK ERR	Error while performing disk GET or ALLOCATE operation
	NO DEFAULT PRIORITY	Priority not specified in INSTALL command and no priority included at task-building time
	FILE NOT FOUND ON LUN 5	Binary file not available for input from LUN-5
	RELOCATION HARDWARE NOT AVA	No relocation hardware available on machine
	FLOATING POINT HARDWARE NOT AVA	No floating-point hardware available on machine
REM	REM-TASK ACTIVE	Task to be removed is currently active
	REM-SYNTAX ERR	Task name omitted
	REM-TASK NOT IN SYSTEM	STL node for task to be removed cannot be found
	REM-DISK ERR	Error whle perfoming disk GET operation
	REM-ALLOCATE ERROR	Error while performing disk ALLOCATE operation
	REM-TASK HAS MULTIPLE STL ENTRIES	Task image is associated with several STL nodes. this condition precludes deallocation of the task disk space.
REQ	REQ-SYNTAX ERR	Task name omitted or priority invalid
	REQ-TASK NOT IN SYSTEM	STL node for requested task cannot be found
	REQ-TASK ALREADY ACTIVE	Requested task is currently active
	REQ-TASK DISABLED	Task has been disabled and is unavailable
	REQ-POOL EMPTY	No nodes left in pool to create new CKQ entry

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	REQ-PART LOST	Partition in which Task is to run has been lost because of reconfiguration
SCH	SCH-SYNTAX ERR	Task name omitted, priority or interval symbol invalid, or time of scheduling and rescheduling noninteger or out of range
	SCH-TASK NOT IN SYSTEM	STL node for scheduled Task cannot be found
	SCH-TASK DISABLED	Task has been disabled and is unavailable
	SCH-PART LOST	Partition in which Task is to run has been lost because of reconfiguration
RUN	RUN-SYNTAX ERR	Task name omitted, priority or interval symbol invalid, or time of activation noninteger or out of range
	RUN-EMPTY POOL	No nodes left in pool to create new ATL entry
	RUN-TASK NOT IN SYSTEM	STL node for activated Task cannot be found
	RUN-TASK DISABLED	Task has been disabled and is unavailable
	RUN-PART LOST	Partition in which Task is to run has been lost because of reconfiguration
SYN	SYN-SYNTAX ERR	Task name omitted, priority or interval symbol invalid, or synchronization unit noninteger or out of range
	SYN-EMPTY POOL	No nodes left in pool to create new CKQ entry
	SYN-TASK NOT IN SYSTEM	STL node for synchronized Task cannot be found

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	SYN-TASK DISABLED	Task has been disabled and is unavailable
	SYN-PART LOST	Partition in which Task is to run has been lost because of reconfiguration
CAN	CAN-SYNTAX ERR	Task name omitted
	CAN-TASK NOT IN SYSTEM	STL node for cancelled Task cannot be found
RES	RES-TASK NOT SUSPENDED	Task to be resumed was not suspended
	RES-SYNTAX ERR	Task name omitted
	RES-TASK NOT ACTIVE	ATL node for resumed Task cannot be found
FIX	FIX-SYNTAX ERR	Task name omitted
	FIX-TASK NOT IN SYSTEM	STL node for fixed Task cannot be found
	FIX-TASK ALREADY FIXED	Task has already been fixed in the partition
	FIX-TASK IS ACTIVE	Task to be fixed is currently active
	FIX-TASK DISABLED	Task to be fixed is currently disabled
	FIX-PARTITION IS OCCUPIED	Partition in which Task is to be fixed is currently occupied
	FIX-EMPTY POOL	No nodes left in pool to create new ATL entry
UNF	UNF-TASK NOT FIXED	Task to be unfixed is not fixed
	UNF-SYNTAX ERR	Task name omitted
	UNF-TASK NOT IN SYSTEM	STL node for Task to be unfixed cannot be found
DIS	DIS-SYNTAX ERR	Task name omitted

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	DIS-TASK NOT IN SYSTEM	STL node for task to be disabled cannot be found
ENA	ENA-SYNTAX ERR	Task name omitted
	ENA-TASK NOT IN SYSTEM	STL node for task to be enabled cannot be found
REA	REA-SYNTAX ERR	Invalid or out-of-range LUN or omitted parameter
	REA-INCORRECT DEV	Invalid device name
	REA-INCORRECT UNIT	Invalid unit on specified device
	REA-INCORRECT LUN	Specified LUN not assigned to associated device
	REA-EMPTY POOL	No nodes left in pool to create new ATL entry
	REA-HANDLER TASK NOT IN SYSTEM	STL node cannot be found for handler of reassigned device
	REA-REASSIGN INHIBITED	REASSIGN cannot be performed at this time, because DISCONNECT AND EXIT or another operation has set the reassign inhibit bit
	REA-OLD DEV NOT YET IN CORE	REASSIGN cannot be performed at this time, because the handler for the device to be reassigned shares its partition with the handler for a busy device
	REA-NO DEFAULT UFD AVAIL	No UFD was given and none is specified in the disk UFD table
	REA-LUN DEDICATED TO MULTIACCESS	REASSIGN is prevented in order to maintain the integrity of the MULTIACCESS system
	REA-DISK GET ERROR	A disk GET or PUT error has occurred
	REA-DISK PUT ERROR	A disk PUT error has occurred
	REA-ALLOCATE ERROR	A disk block could not be allocated to initialize a UFD

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Messages	Possible Meaning
SAV	TASKS ACTIVE -- SAVE ERROR	SAVE cannot be performed while entries (active tasks) remain in the ATL
	SAV-I/O REQUESTS PENDING	System not quiescent because of pending I/O requests
	SAV-STL NOT ENTIRELY RECONSTRUCTED	On bootstrapping, the STL could not be completely rebuilt, because of empty pool of nodes
	SAV-DISK PUT ERROR	Error while performing disk PUT operation
	SAV-DISK GET ERROR	Error while performing disk GET operation
	SAV-NO SAVE AREA ON SYSTEM DISK	Unlikely error resulting from reassigning system device
	SAV-RELOCATION NECESSARY	Relocation hardware necessary and not available
	SAV-PROTECT-RELOCATE SWITCH MUST BE IN RELOCATE POSITION	Switch is in wrong position for relocation
	SAV-FLOATING-POINT UNIT NECESSARY	Floating-point hardware necessary and not available on machine
	SAV-REENTRANT ECO PACKAGE NECESSARY TO RUN RSX	Reentrant ECO package necessary and not availble on machine
OPE	OPE-SYNTAX ERR	Invalid block, register or offset
	OPE-DSK ERR	Error while perfoming disk GET operation
ABO	ABO-SYNTAX ERR	Task name omitted
	ABO-TASK NOT IN SYSTEM	STL node for task to be aborted cannot be found
	ABO-TASK INACTIVE	ATL node for task to be aborted cannot be found
	ABO-SYSTEM TASK	It is illegal to abort BATCH or the MULTIACCESS Monitor
ADV	FORMAT ERROR	Task name omitted or invalid device name
	DEVICE EXISTS ALREADY	PDVL node for the specified device already exists

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	UNIT NUMBER IS OUT OF ORDER	Unit number specified in ADV command not sequential with current unit numbers
	POOL IS EMPTY	No nodes left in pool to create new PDVL entry
	UNITS OF THIS DEVICE ARE ASSIGNED	No available unit numbers for specified device name
ASP	FORMAT ERROR	Partition or Task name omitted or invalid
	CAN NOT FIND PBDL NODE	PBDL node for specified partition cannot be found
	CAN NOT FIND STL NODE	STL node for specified Task cannot be found
	EXEC MODE TASKS CANNOT BE RELOCATED	Partition for EXEC-mode Task cannot be changed
	TASK IS ACTIVE	Task whose partition is changed is currently active
	TASK IS FIXED IN CORE	Task is currently fixed in a core partition
	PARTITION IS TOO SMALL	Partition to which Task is being assigned is not large enough
	DISABLE ERROR	Error while disabling Task during partition reassignment
	ENABLE ERROR	Error while reenabling disabled Task in new partition
MNT	ALLOCATION ERROR	Error while performing disk allocate operation
	DISK PUT ERROR	Error while performing disk put operation
	FORMAT ERROR	Invalid device name, unit, or UIC

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	DISK HAS NO PDVL NODE	PDVL node for specified disk cannot be found
	DEVICE IS NOT A DISK	Device name does not correspond to a disk in the RSX system
	ILLEGAL TO MOUNT THE SYSTEM DISK	Device name corresponds to the system disk
	DISK NOT DISMOUNTED	Specified disk has not been dismounted since last mount
	DISK GET ERROR	Error while performing disk get operation
DSM	FORMAT ERROR	Invalid device name or unit
	DISK HAS NO PDVL NODE	PDVL node for specified disk cannot be found
	DEVICE IS NOT A DISK	Device name does not correspond to a disk in the RSX system
	ILLEGAL TO DISMOUNT THE SYSTEM DEVICE	Device name corresponds to the system disk
	DEVICE NOT MOUNTED	Disk to be dismounted is not currently mounted
	DEVICE IS IN USE	Specified disk is currently being used for I/O
	MARK TIME ERROR	Error encountered while marking time until all open files are closed
DEQ	ILL PARAM	Invalid DEQ symbol
	OUTPUT ERROR	Error while trying to list deque on LUN-16 device (normally line printer)
RCF RCP	OVERFLOW	Partition specified does not fit in available core area or core being used as system-reserved area
	SYNTAX ERR AT x	Invalid character or delimiter encountered at x
	INVALID SIZE	Base or size specification invalid

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	NAME USED	Name assigned to partition, COMMON block, etc. already in use
	SYS NAME	Reserved system name assigned
	SYS SPACE	Attempt to allocate in area reserved for system use
	CORE IN USE	Task active in core area specified for new partition
	OUT OF LARGE NODES	No more large nodes can be found in pool
	OUT OF SMALL NODES	No more small nodes can be found in pool
	FATAL SYS ERROR	Serious system or RCF malfunction
	SYS BLK>32K	Nodes have been specified to occupy space above 32K or a memory block extends above 128K.
SLI	FORMAT ERROR	Omitted priority, invalid ticks, or invalid priority range (e.g., low above high)
	ILLEGAL PRIORITY	Noninteger or invalid priority
	TIME SLICING IS ALREADY ON	SLICE OFF command has not been given to negate previous time-slicing

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	TICKS MUST BE NON-ZERO	Number of ticks omitted or zero supplied
SHU	SHU - BATCH ACTIVE	SHUT operation cannot be performed at this time, because a batch job is running
CON	SYNTAX ERROR	Task name or LUN omitted, or invalid or out-of-range LUN
	CREATE ERR	Error while performing CREATE operation
	READ ERR	Error while performing READ operation

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
	DISK ERR	Error while performing disk GET or ALLOCATE operation
	FILE NOT FOUND	Binary file not available on LUN-5
XQT	EXE-SYNTAX ERR	Task name of LUN omitted or invalid partition or priority
	EXE-TASK NOT IN SYSTEM	STL node for task to be executed cannot be found or FININS not installed
	PARTITION LOST THROUGH RCF	Partition in which task is to be executed has been lost because of reconfiguration
	EXE-TASK DISABLED	Task has been disabled and is unavailable
	EXE-POOL EMPTY	No nodes left in pool to create a new ATL entry
DOS	DEALLOCATION ERROR	Error while performing DEALLOCATE operation
	ILLEGAL SYSTEM DEVICE	Device assigned as system device is invalid for DOS use
	TASK ERROR	Error while performing disk READ operation
ACI	FATAL ERROR -- EXIT	I/O error while attempting to access account file; event variable follows message
	INVALID -- EXIT	Incorrect password supplied
ACD	***FATAL ERROR ACCESSING ACCOUNT FILE***	I/O error while attempting to access account file; event variable follows message

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
DTC	FORMAT ERROR	The command line contains a typographical error
	ILLEGAL UNIT NUMBER	The terminal handler has no record of the existance of this unit
	TERMINAL BUSY	The terminal has an outstanding READ or WRITE request
	ILLEGAL BAUD RATE	The baud rate specified is illegal for the device

(Continued on next page)

Table 3-1 (Cont.)
MCR Error Messages

Task	Error Message	Possible Meaning
STR	STROBE--BUFFER OVERFLOW*	The list specified is too long to fit in STROBE's internal buffer
	STROBE--FORMAT ERROR	The command string contains a typographical error
	STROBE--ILLEGAL LISTHEAD	This LISTHEAD is 0, or it's not within the 1st 32K or core, or if the core size is less than 32K the listhead is not within the core size specified by SCOM register 136.
	STROBE--OUTPUT ERROR	STROBE has detected an output error on LUN-16
	STROBE--ILLEGAL POINTER*	One of the nodes in the list has an illegal forward pointer
ACC	ACC--FORMAT ERROR	The command string contains a typographical error
	ACC--MEMORY BLOCK NOT FOUND	The memory block named does not exist.
	ACC--MEMORY BLOCK IN USE	The memory block is in use

*This is not a fatal error, however, STROBE will print as much of the list as possible.

INDEX