XVM/RSX PART V
SYSTEM DIRECTIVES

CHAPTER 1

INTRODUCTION TO SYSTEM DIRECTIVES

1.1 FUNCTION OF SYSTEM DIRECTIVES

RSX user Tasks carry on communication with the Executive by means of System Directives which perform such functions as scheduling and rescheduling Task execution, suspending and resuming a Task, connecting to an interrupt line, and fixing a Task in core. Most Directives can be issued in any of the following ways:

- by the user from within a FORTRAN IV Task
- by the user from within a MACRO Task
- by the operator via the Monitor Console Routine

1.2 DESCRIPTION OF DIRECTIVES

System Directives are implemented as CAL instructions. These are machine language instructions that transfer control from anywhere in core to the monitor via fixed-memory locations. Each CAL instruction takes a parameter block as an operand: the instruction's address field points to the first word of the associated block of parameters. This block consists of one or more words of contiguous core and is called a CAL Parameter Block (CPB).

Most System Directives can be executed directly from MACRO programs. When a System Directive is issued by a FORTRAN program, the program CALLs one of a series of FORTRAN Library Routines; the routine then issues the Directive. Monitor Console Routine (MCR) Function Tasks issue direct calls for specific System Directives. Exact procedures for invoking System Directives are discussed in subsequent sections of this manual. Table 1-1 contains a list of all System Directives implemented under RSX.

1.3 PROCESSING A DIRECTIVE

When RSX processes a system directive, control is passed to a reentrant CAL Dispatch Routine. This routine takes the low-order six bits of the first CPB word as an octal CAL function code and, using this code as an index, transfers control to a particular CAL service routine. After the CAL service routine has performed the operation requested by the directive, it passes control to a CAL Exit Routine, which returns control to the directive-issuing task. Execution of this task continues at the location immediately following the CAL instruction.

NOTE

> To minimize Monitor overhead, registers such as AC, MQ and XR are neither saved nor restored when a CAL instruction is executed.

The CAL Dispatch Routine and many CAL service routines are reentrant. This feature allows an interrupt service routine to issue Monitor requests without waiting for completion of a request issued by a task.

Most system directives take an event variable in the MACRO or FORTRAN call. With few exceptions, inclusion of this event variable is optional. If an event variable is specified, a completion code indicating the status of the directive is returned to the issuing task. A code of zero means that the directive is pending, a positive code means that it has been completed successfully, and a negative code means either that it has not been accepted or that it has been completed, but an error has occurred. The value of the negative code indicates the particular fault.

1.4 DIRECTIVE ERRORS

Before attempting to perform the operation requested by the directive call, the CAL service routine checks for a variety of serious errors. Following are possible errors it detects during this initial check:

1. Executing a CAL, such as XCT (CAL CPBADR).

2. CPB with illegal function code (one not included in Table 1-1).

3. Event variable address outside the task area. This check is made only for tasks built in user mode.

4. CPB address outside the task area. This check is made only for tasks built in user mode.

In general, verification checks on address parameters are performed for user-mode tasks, but not for exec-mode tasks. Other errors may be detected during actual performance of the operation. The number of ignored CALs is logged as a performance indicator or debugging aid in SCOM location 151 (octal) in the Executive.

If one of the above errors is detected, the issuing task is terminated (forced to exit) and a task termination notice with the following format is printed:

```
BAD CAL AT PC = Xyyyyy
```

In this format, Xyyyyy is an 18-bit word in which X identifies the task status with the following three bits:

Bit	Meaning
0	Contents of link
1	Addressing mode: 0 = page mode; 1 = bank mode
2	Protection mode: 0 = exec mode; 1 = user mode

In the above format, yyyyy is a 15-bit address indicating the location of the error. For user-mode tasks, this address is relative to the partition base.

For tasks requested on behalf of a MULTIACCESS user, the termination notice is printed on the user terminal. For all other tasks, the message is sent to the device assigned to system LUN-3.

The MULTIACCESS Exit Processor controls the printing of abnormal task termination messages for all tasks run under control of that Monitor. To cause printing of this task termination notice for all other tasks, an entry is made in the Task Termination Notice Request List (TNRL) and a disk-resident task named TNTERM is requested by the system.

1.5 DIRECTIVE CONVENTIONS

System directives are constructed according to conventions that differ slightly for MACRO and FORTRAN calls but basically consist of the following:

1. The system directive name is followed by a space in the calling line.
2. All parameters included in the call are separated by commas.
3. FORTRAN parameters are enclosed in parentheses.
4. Rules about line termination and error correction conform to MACRO or FORTRAN program standards.

5. In "form" models, upper-case characters (except LUN) indicate those entries required by the system. Lower-case characters indicate entries that are to be specified by the user and are dependent on the particular task.

6. "MACRO call" refers to entries in the MACRO definition file (RMC.13 SRC). This file is usually not used when assembling MACRO programs, because the MACRO definitions increase the size of the resulting binary file.

Table 1-1 summarizes legal calls to all RSX system directives.

Table 1-1
System Directives

CAL Code (octal)	MACRO Call	FORTRAN Call	MCR Call	Internal
00	I/O calls	I/O calls	---	---
01	REQUEST	CALL REQST	REQ[UEST]	---
02	SCHEDULE	CALL SCHED	SCH[EDULE]	---
03	RUN	CALL RUN	RUN	---
04	CANCEL	CALL CANCEL	CAN[CEL]	---
05	WAIT	CALL WAIT	---	---
06	SUSPEND	CALL SUSPND	---	---
07	RESUME	CALL RESUME	RES[UME]	---
10	EXIT	CALL EXIT	See ABORT	---
11	CONNECT	---	---	---
12	DISCONNECT	---	---	---
13	MARK	CALL MARK	---	---
14	SYNC	CALL SYNC	SYN[C]	---
15	FIX	CALL FIX	FIX	---
16	UNFIX	CALL UNFIX	UNF[IX]	---
17	---	---	---	SETJEA
20	WAITFOR	CALL WAITFR	---	---
21	DISABLE	CALL DISABL	DIS[ABLE]	---
22	ENABLE	CALL ENABLE	ENA[BLE]	---
23	UNMARK	CALL UNMARK	---	UNMARK
24	DATE	CALL DATE	DAT[E]	---
25	---	---	---	TSKNAM
26	---	---	---	PARINF
27	---	---	---	RAISEB
30	---	---	---	RASP
31	---	CALL SPY	---	---
32	---	CALL SPYSET	---	---
33	---	CALL QJOB	---	---
34	---	CALL EXECUT	XQT	---
35	---	CALL SHARE	---	---
36	---	---	---	REQMAP
37	---	---	---	XFRCMD

CHAPTER 2

SYSTEM DIRECTIVES

Each of the following sections describes one System Directive. In all models and examples, the following conventions apply:

1. A space in the text indicates an actual space in the Directive call.
2. Square brackets ([]) indicate optional parameters.
3. In "Form" models, upper case characters indicate those required by the system, while lower case characters indicate entries which are to be specified by the user and are dependent on his particular Task.

2.1 CANCEL: CANCELLING REQUESTS FOR A TASK

The CANCEL Directive instructs RSX to CANCEL all scheduled requests (Clock Queue entries made by SCHEDULE, RUN, or SYNC) for activation of a particular Task by nullifying those requests in the Clock Queue. CANCELlation affects neither future scheduling of a Task (as DISABLE does), nor its current execution. The user can optionally include an Event Variable in the Directive. Note that Task CANCELlation does not remove any nodes from the Clock Queue. Nodes are removed only when the time given in the node comes due.

CANCEL Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. A Task cannot CANCEL itself. The MACRO call has the following form:

Form:	CANCEL name[,ev]
Where:	name of Task CANCELled is a string of one to six .SIXBT characters ev is the Event Variable address
Example:	CANCEL SCAN,EV

The FORTRAN call has the following form:

Form:	CALL CANCEL(nHname[,ev])
Where:	n is the number of characters in name name of Task CANCELled is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL CANCEL(4HSCAN,IEV)

The CANCEL Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (04)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system

If the CANCEL is accepted, the Event Variable (if specified) is set to +1, and all scheduling nodes in the Clock Queue for this Task are nullified. Following is an example of a CANCEL CPB:

```
CPB 04                  /CAL PARAMETER BLOCK TO CANCEL
    EV                  /ALL SCHEDULE REQUESTS FOR THE
    .SIXBT   "T7@@@@"   /TASK "T7".
```

2.2 CONNECT: CONNECTING TO AN INTERRUPT LINE

The CONNECT Directive instructs RSX to transfer control to a particular location (entry point) whenever an interrupt occurs on an indicated interrupt line. In effect, this creates a link between a specific Automatic Priority Interrupt (API) trap address and a specific entry point to an interrupt service routine. One trap address exists for each of the 32 API lines supplied on the computer. The following list does not include device lines that are always CONNECTed to the system (e.g., Clock, Disk, and Teletype keyboards and printers). Line numbers above 37 octal are pseudo-API lines, simulated by the Executive for devices connected only to the Program Interrupt (PI).

The user can optionally include an Event Variable in the Directive.

Additional DECtape (line 36) or an additional Dataphone (line 37) cannot be added without modifications to the Handlers for these devices.

CONNECT Directives can be issued by MACRO programs in the following way:

Form:	CONNECT line,entry[,ev]
Where:	line is an octal number representing the interrupt line to be CONNECTed entry is the entry address of the related interrupt service routine ev is the Event Variable address
Example:	CONNECT an interrupt service routine for an A/D Converter at entry point ADINT to interrupt line 17: CONNECT 17,ADINT,ADEV

Because FORTRAN is not an appropriate language for writing interrupt-handling routines, no FORTRAN call is included. Connects cannot be issued from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (11)
1	Event Variable address
2	Interrupt line number
3	Interrupt transfer address

Table 2-1
API Lines

Octal Line Number	Device
00 +40	Software level 4
01	Software level 5
02	Software level 6
03	Software level 7
04	DECtape
05	Magtape
06	(Unused)
07	Disk Cartridge (RK-UC15 Unichannel)
10	Paper Tape Reader
11	Clock overflow
12	Power failure
13	Memory parity error
14	Display (and reserved for VP storage scope)
15	Card Reader
16	Line Printer
17	Analog-to-Digital Converter
20	DB99A-DB98A Interprocessor Buffer
21	(Unused)
22	Dataphone
23	Fixed-head disk (RF)
24	Disk Pack (RP)
25	XY Plotter
26	(Unused)
27	(Unused)

(Continued on next page)

Table 2-1 (Cont.)
API Lines

Octal Line Number	Device
30	(Unused)
31	(Unused)
32	(Unused)
33	(Unused)
34	LT19 Output
35	LT19 Input
36	Additional DECtape
37	Additional Dataphone
40	Console Teletype Printer
41	Console Teletype Keyboard
42	Paper Tape Punch
43	Memory-protect violation

If the CONNECT Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-26	Illegal Function for a USER-mode Task
-301	Illegal line number
-302	Indicated line already CONNECTed

If the CONNECT Directive is accepted, the Event Variable (if specified) is set to +1 and the CONNECTion is established.

Following is an example of a CONNECT CPB:

```
CPB 11      /CAL PARAMETER BLOCK TO CONNECT THE
    EV      /DECTAPE INTERRUPT (LINE #4) TO THE
    4       /LOCATION "DTINT".
    DTINT
```

2.3 DATE: RETRIEVING TIME AND DATE

The DATE Directive instructs RSX to return the current date and time to the issuing Task. This information is obtained from the system's internal clock and calendar. The Directive can be issued by MACRO and FORTRAN programs and by interrupt service routines. The user can optionally include an Event Variable in the Directive. Following is the MACRO call:

Form:	DATE month,day,year,hour,minute,second[,ev]
Where:	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 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 ev is the Event Variable address
Example:	DATE MON,DAY,YR,HR,MI,SEC,EV

and the FORTRAN call:

Form:	CALL DATE(d)
Where:	d is an array consisting of six integer words: d(1) = month in range 1-12 decimal d(2) = day in range 1-31 decimal d(3) = year in range 1-99 decimal; represents the last two digits in the calendar year d(4) = hour in range 0-23 decimal d(5) = minute in range 0-59 decimal d(6) = second in range 0-59 decimal
Example:	DIMENSION IDATE(6) . . . CALL DATE(IDATE)

This Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (24)
1	Event Variable address
2	Month buffer (1-12)
3	Day buffer (1-31)
4	Year buffer (0-99)
5	Hour buffer (0-23)
6	Minute buffer (0-59)
7	Second buffer (0-59)

The Event Variable (if specified) is always set to +1. The date and time are stored in CPB words 2 through 7.

Following is an example of a DATE CPB:

```
CPB 24          /CAL PARAMETER BLOCK
    EV          /TO OBTAIN DATE
    .BLOCK 6    /AND TIME.
```

2.4 DISABLE: DISABLING A TASK

The DISABLE Directive instructs RSX to reject all future REQUEST, SCHEDULE, RUN, FIX, and SYNC Directives for a particular Task. It does not have the effect of CANCEL, for it does not affect current activity: if rescheduling is already in effect, the Clock Interrupt Service Routine periodically requests the Task. If a Task has been DISABLEd, it is capable of responding only to the ENABLE Directive. If the Task is active when the DISABLE is issued, it continues to execute; however, all subsequent requests for the Task are ignored. A DISABLEd Task is not automatically deleted from the system. Only the REMOVE function can do this. The user can optionally include an Event Variable in the Directive.

DISABLE Directives can be issued by MACRO and FORTRAN programs and by interrupt service routines. Following is the MACRO call:

Form:	DISABLE name[,ev]
Where:	name of Task DISABLEd is a string of one to six .SIXBT characters ev is the Event Variable address
Example:	DISABLE SCAN,EV

and the FORTRAN call:

Form:	CALL DISABL(nHname[,ev])
Where:	n is the number of characters in name name of Task DISABLEd is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL DISABL(4HSCAN,IEV)

The DISABLE Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (21)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the DISABLE Directive is rejected by RSX, the Event Variable (if specified) is set to the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system

If the DISABLE Directive is accepted, the Event Variable (if specified) is set to +1 and the Task is DISABLEd.

Following is an example of a DISABLE CPB:

```
CPB 21                  /CAL PARAMETER BLOCK
    EV                  /TO DISABLE TASK "T11".
    .SIXBT "T11@@@"
```

2.5 DISCONNECT: DISCONNECTING FROM AN INTERRUPT LINE

The DISCONNECT Directive instructs RSX to nullify a CONNECT. Thus, when an interrupt occurs on an indicated interrupt line, RSX no longer transfers control to an indicated location (entry point) of a particular interrupt service routine. The I/O Handler for the relevant device ensures that all I/O transfers have been completed before the DISCONNECT is accepted. The user can optionally include an Event Variable in the Directive.

DISCONNECTs can be issued by MACRO programs in the following way:

Form:	DISCONNECT line,entry[,ev]
Where:	line is an octal number representing the connected interrupt line entry is the entry address of the related interrupt service routine ev is the Event Variable address
Example:	DISCONNECT the A/D Converter from interrupt line 17: DISCONNECT 17,ADINT,ADEV

Because FORTRAN is not an appropriate language for writing interrupt-handling routines, no FORTRAN call is included. DISCONNECTs cannot be issued from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (12)
1	Event Variable address
2	Interrupt line number
3	Current interrupt transfer address

If the DISCONNECT Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-26	Illegal function for a USER-Mode Task
-301	Illegal line number
-302	Line not connected as indicated

If the DISCONNECT Directive is accepted, the Event Variable (if specified) is set to +1 and the DISCONNECT is performed.

Following is an example of the DISCONNECT CPB:

```
CPB 12      /CAL PARAMETER BLOCK TO DISCONNECT
    EV      /LINE #4 (DECTAPE) FROM "DTINT".
    4
    DTINT
```

2.6 ENABLE: REENABLING A TASK

The ENABLE Directive instructs RSX to reENABLE a disabled Task. The user can optionally include an Event Variable in the Directive.

This Directive can be issued by MACRO and FORTRAN programs and by interrupt service routines. Following is the MACRO call:

Form:	ENABLE name[,ev]
Where:	name of Task ENABLEd is a string of one to six .SIXBT characters ev is the Event Variable address
Example:	ENABLE SCAN,EV

and the FORTRAN call:

Form:	CALL ENABLE(nHname[,ev])
Where:	n is the number of characters in name name of Task ENABLEd is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL ENABLE(4HSCAN,IEV)

The ENABLE Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (22)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the ENABLE Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured

If the ENABLE Directive is accepted, the Event Variable (if specified) is set to +1 and the Task is ENABLEd.

Following is an example of an ENABLE CPB:

```
CPB 22                  /CAL PARAMETER BLOCK
    EV                  /TO ENABLE TASK "T12".
    .SIXBT "T12@@@"
```

2.7 EXECUTE: REQUESTING TASK EXECUTION FROM A USER DISK

The EXECUTE directive allows the user to request execution of a task that is stored in core-image form on a "user" disk.

A new task priority can be specified in the EXECUTE directive. If a new priority is omitted, the priority specified at task-building time is used.

As options, a memory partition and an event variable can be specified in the directive. If the directive is issued by a MACRO program, two additional options can be specified: "alias execute" and "deferred execute".

The standard EXECUTE directive functions internally by:

1. Inserting a node in the Execute List (EXELH) for the requested task.
2. Requesting the task named FININS.

FININS:

1. Picks a node off of the EXELH. If the list is empty, the EXECUTE directive exits.
2. Locates a created file with the name specified in the EXECUTE command in the disk directory associated with the specified LUN.
3. Requests the task (if its node is in the STL) or allocates sufficient space on the system disk and transfers the file to the system disk.
4. For exec-mode tasks, ensures that the specified partition name corresponds to the partition and base address indicated for the task at task-building time. 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 to ensure that the task image fits in the partition.
5. Enters a node in the STL for the task, setting a "remove-on-exit" bit and a bit indicating that the task has been run at least once ("done" bit).

Periodically, a task called AUTORM checks the STL and removes all tasks from the system disk that:

a. Are not active

b. Have the "done" bit set

c. Have the "remove-on-exit" bit set

This periodic check and removal allows substantial conservation of space on the disk. If, however, a task is running or has been scheduled by a directive such as SCHEDULE or SYNC, AUTORM does not remove it until a later check.

The events listed to this point are modified slightly by selecting the deferred-execute option.

If the EXECUTE directive is rejected by RSX, the event variable (if specified) is set to one of the following values to indicate rejection and the cause:

Event Variable	Meaning
-201	Task is not in the system
-202	Task is already or still active
-204	Task is disabled
-206	Illegal task priority
-212	Partition for the task STL node was lost through reconfiguration
-213	Partition assigned to the task is currently being reconfigured
-777	Deque node (for ATL) is not available (POOL empty)

If the directive is accepted, the event variable (if specified) is set to +1.

EXECUTE directives may be issued by FORTRAN and MACRO programs, and can also be called from the Monitor Console Routine (refer to Part IV of this manual).

The FORTRAN call is of the following form:

Form:	CALL XQT(nHname,p,LUN,mHpartnam[,ev])
Where:	n is the number of characters in the task name name of task EXECUTEd is a string of one to five ASCII characters p is an integer priority in decimal range 0-512 LUN is an integer representing the logical unit number associated with the user disk m is the number of characters in partition name partnam is the name of the partition in which the task should be run ev is the integer event variable
Example:	Request EXECUTion of TTIN in partition TDV at priority 20: IP = 20 CALL XQT (4HTTIN,IP,8,3HTDV,IP,IEV)

The requested task is made active at priority P (or at the default priority, if p is omitted or zero). If a task with mapped LUNs (such as a MULTIACCESS task) issues an EXECUTE directive, the virtual LUN specified is mapped into the corresponding system LUN.

The CAL parameter block (CPB) for this directive consists of the following:

Word	Contents
0	CAL function code (34)
1	Event variable address
2	Task name (first half)
3	Task name (second half)
4	Priority
5	LUN, "alias execute" bit and "deferred execute" bit
6	Partition name (first half) or task event variable address
7	Partition name (second half)
10	Unused or secondary task name (first half)
11	Unused or secondary task name (second half)

The alias-execute option is specified by setting bit 0 of CPB word 5, which contains the LUN. When this option is specified, words 10 and

11 of the CPB are taken to be a secondary task name. FININS replaces the primary task name with the secondary task name when it prepares a node for insertion into the STL. Any time after the node is inserted into the STL, FININS requests the task using the secondary task name. FININS always uses the primary task name when searching disk file directories for the constructed file. However, if the alias-execute option is specified, FININS first scans the STL using the secondary task name to see if the file is already installed in the system.

The deferred-execute option is specified by setting bit 1 of CPB word 5. When this option is specified, word 6 of the CPB is taken to be the address of a special "task" event variable instead of the first half of a partition name. FININS sets the task event variable to the address of the task STL node. When creating the task STL node, FININS zeros the pointer to the task PBDL node (word 5) and sets the "partition-lost-through-reconfiguration" bit (bit 4 of word 4). FININS attempts to locate and transfer the task image to the system disk as usual, but does not request the task.

The result of these actions by FININS is to prepare a task for execution later when an appropriate partition has been selected. This feature is used primarily by MULTIACCESS.

If FININS detects an error, it sets the task event variable to one of the following values:

Task Event Variable	Meaning
-401	Illegal LUN specified
-402	HINF error
-403	Illegal device specified
-404	Disk dismounted
-405	GET error
-406	File was not created
-407	File was not found
-410	ALLOCATE error
-411	PUT error
-412	Nonunique alias name

If FININS is successful, the task event variable (if specified) is set to +1.

2.8 EXIT: TERMINATING EXECUTION OF A TASK

The EXIT directive instructs RSX to terminate execution of the task that issues the EXIT. If the issuing task is not fixed in core, the core partition of this task becomes available to other tasks. For this reason, an EXIT should not be issued until all transfers to or from the partition (e.g., I/O transfers, task-to-task transfers, event variable settings) have been completed. No event variable can be included in the EXIT directive.

RSX provides protection against premature task EXITs. When any task exits, the EXIT Routine checks the transfers-pending count in the task partition block node. A nonzero count indicates that the EXITing task has pending I/O or mark-time requests. This causes I/O Rundown to be invoked. As a result of this operation, the partition in which the EXITing task resides does not become free until transfers to or from the partition have stopped. ●

The EXIT directive can record the amount of processing time used by the task that is exiting, provided that the system has XM15 hardware and the proper software initialization has occurred. To use this feature, the user must develop an exec-mode task that is capable of performing the necessary initialization and dump data buffers to a mass-storage peripheral device. The steps required to activate the task timing software are:

1. Zero each location in a pair of data buffers. These buffers can be anywhere in core, but must be at least four words long and should typically have a size that is an integral multiple of four words.

2. Set up a task timing control table as follows:

Word	Name	Contents
0	EV	Task timing event variable (must be zero initially).
1	START1	Pointer to first word of buffer 1.
2	END1	Pointer to last word of buffer 1.
3	START2	Pointer to first word of buffer 2.
4	END2	Pointer to last word of buffer 2.
5	PTR	Pointer to next free buffer entry (set equal to START1 initially). The task timing software does no error checking to ensure the validity of the parameters in this control table. It is the user's responsibility to supply the correct data.

3. Set absolute location 312 (hereafter referred to as TIMFLG) to the address of the task timing control table. The task timing routine in the EXIT directive uses TIMFLG as a flag to indicate if task timing is to be invoked. Whenever a task EXITs, TIMFLG is examined. If it is zero, the task timing routine will be bypassed; but if TIMFLG is non-zero, EXIT will assume that the task timing routine should be entered. Since the user-written buffer-dumping initialization task can never tell when EXIT will examine TIMFLG, this task should inhibit interrupts anytime it manipulates that location to prevent a race condition.

   When TIMFLG is set, a task EXITs, and there is sufficient room in a buffer, the task timing routine will fill the appropriate buffer entry with the following data:

ENTRY WORD	CONTENTS
0	First half of the task name in .SIXBT format
1	Second half of the task name in .SIXBT format
2	Number of XM clock overflows (one unit equals 2.62 seconds)
3	Number of XM clock ticks above overflow count (one unit equals 10 microseconds)

If neither buffer is full, the task timing event variable (EV) will remain zero. However, EXIT will typically set EV to +1 when buffer 1 if filled and +2 when buffer 2 is filled. EXIT assumes that when the buffer-dumping task has dumped a buffer and reinitialized it to zeroes, that task will indicate that the buffer is empty by zeroing EV. To avoid a race condition when the buffer-dumping task is manipulating EV, it must inhibit interrupts.

If EXIT determines that both buffers have been filled, it will set EV to -1. At this point, no task timing data will as yet have been lost. If another task EXITs before a buffer is emptied, EXIT will decrement the already negative EV. The buffer-dumping task must then empty a buffer, record the number of data entries lost, and set EV to the number of the buffer still full. To stop task timing, the buffer dumping task need only zero TIMFLG.

Since the task timing routine can only be invoked when a task EXITs, some tasks will seldom, if ever, be timed. Tasks in this category include DSK, DSA, IORD, and semipermanent I/O handlers such as RF, RK, and RP. Hence, if task timing data is needed for these tasks, the user must implement some special code to obtain the required data whenever the system is SAVEd and whenever DOS is bootstrapped via the relevant MCR Function Tasks.

EXIT Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. A CPB exists for EXIT, but a standard MACRO call has not been implemented. The CAL is issued as follows:

```
        CAL (10)        /TASK EXIT--TERMINATE EXECUTION
```

A FORTRAN STOP command can be used to terminate execution of a FORTRAN TASK.

EXITs apply only to calling Tasks. Task exit can be forced via the Monitor Console Routine by use of the ABORT function.

The CPB for this Directive consists simply of the following:

<u>Word</u>	<u>Contents</u>
0	CAL function code (10)

2.9 FIX: FIXING A TASK IN CORE

The FIX Directive instructs RSX to FIX an inactive Task in an available core partition. This dedicates the partition to that Task and provides for faster response to REQUEST, SCHEDULE, RUN, and SYNC Directives. FIX does not cause a Task to be executed at the time the Directive is issued. The user can optionally include an Event Variable.

This Directive can be issued by MACRO and FORTRAN programs and also by interrupt service routines. FIX cannot be issued to an active Task, but a Task can FIX itself. In this case, the Directive causes the Task to retain control of its partition even after it exits. Following is the MACRO call:

Form:	FIX name[,ev]
Where:	name of Task FIXed is a string of one to six .SIXBT characters ev is the Event Variable address
Example:	FIX SCAN,EV

and the FORTRAN call:

Form:	CALL FIX(nHname[,ev])
Where:	n is the number of characters in name name of Task FIXed is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL FIX(4HSCAN,IEV)

The FIX Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (15)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the FIX Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system
-202	Task is active
-204	Task is disabled
-207	Task already FIXed in core
-210	Partition occupied
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured
-777	Deque node (for PBDL) unavailable (empty pool)

If the FIX Directive is accepted, the Event Variable (if specified) is set to +1 and the Task is flagged as being FIXed in core. The FIXed Task is automatically made active at status one to effect loading into core. The starting address of this Task is an EXIT Directive in the Executive; therefore, the Task exits after it is loaded into core, and subsequent EXIT Directives do not free the FIXed partition. Only UNFIX can do this.

Following is an example of a FIX CPB:

```
CPB 15                  /CAL PARAMETER BLOCK TO
    EV                  /FIX TASK "T9" IN CORE.
    .SIXBT "T9@@@@"
```

2.10 MARK: SETTING AN EVENT VARIABLE IN THE FUTURE

The MARK Directive instructs RSX to set an Event Variable to zero and, after a specified interval of time, to reset it to a nonzero value and to declare a Significant Event.

Table 2-2
Reschedule Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	1	0-262143
Second	2	0-86400
Minute	3	0-1440
Hour	4	0-24

This Directive can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. MARK Directives affect only the Tasks from which they are issued. Following is the MACRO call:

Form:	MARK mi,mu,ev
Where:	mi is an integer representing the number of units which must elapse before the Event Variable is set nonzero (range given in Table 2-4) mu is 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-4) ev is the Event Variable address
Example:	Reset Event Variable TSTEV 5 minutes from now: MARK 5,3,TSTEV

and the FORTRAN call:

Form:	CALL MARK(t,ev)
Where:	t is an array consisting of two integer words representing the MARK time interval and unit: t(1) = MARK time interval (range given in Table 2-4) t(2) = 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-4) ev is the integer Event Variable
Example:	Reset Event Variable IEV 5 minutes from now: DIMENSION IT(2) . . . IT(1) = 5 IT(2) = 3 CALL MARK(IT,IEV)

The MARK Directive cannot be issued from the Monitor Console Routine.

The CPB for this directive consists of the following:

Word	Contents
0	CAL function code (13)
1	Event Variable address
2	MARK-time interval (see Table 2-4)
3	MARK-time units (see Table 2-4)

If the MARK Directive is rejected by RSX, the Event Variable is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-203	Directive not Task-issued
-777	Deque node (for Clock Queue) not available (empty pool)

If the MARK Directive is accepted, the Event Variable is set to zero, and a request to reset the Event Variable is entered in the Clock Queue to come due at the appropriate time. Issuing an UNMARK Directive can nullify a request.

Following is an example of a MARK CPB:

```
          CPB 13       /CAL PARAMETER BLOCK TO CLEAR "EV"
              EV       /AND SET IT TO NONZERO 170 SECONDS
              .DEC     /FROM NOW.
              170; 2
              .OCT
```

2.11 PARINF: RETURNING PARTITION ADDRESS AND SIZE

The PARINF Directive instructs RSX to return the base address and size of the core partition named in the Directive to the issuing Task. If a partition is not specified (if CPB word 2 is zero), RSX references the name of the partition in which the issuing Task runs and stores it in CPB words 2 and 3. PARINF Directives are ordinarily not issued by user-written code. Therefore no FORTRAN call or standard system MACRO has been implemented for PARINF.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (26)
1	Event Variable address
2	Partition name (first half)
3	Partition name (second half)
4	Address of two-word information buffer

The partition name is coded in .SIXBT form.

The information buffer referenced in word 4 contains:

Word	Contents
0	Partition base address
1	Partition size (octal)

If a PARINF Directive cannot be honored, one of the following Event Variables is returned to indicate rejection and its cause:

Event Variable	Meaning
-203	CAL not Task-issued
-211	Named partition not in system

If the PARINF Directive is accepted, the Event Variable is set to +1 and the partition information is returned.

Following is an example of a PARINF CPB:

```
          CPB     26                  /FIND OUT WHERE
                  EV                  /"TDV" PARTITION
                  .SIXBT "TDV@@@"     /IS AND STORE
                  PARINF              /THE DATA
          /
          PARINF  .BLOCK 2            /HERE.
```

2.12 QJOB: QUEUEING A BATCH JOB

The QJOB directive informs the Batch System that a job is ready to be run, whether or not the Batch Processor is in core. The user can specify the name of the job to be queued, the LUN from which it comes and a series of job characteristics, including:

- Maximum time that the job can run (in minutes)
- Class at which the job can run
- Memory use
- Use of sequencing (run in order of submission)
- Whether the job requires operator availability
- Whether the job file should be deleted after the job runs
- Use of hold mode
- Device and UFD of the job file

The FORTRAN call for QJOB has the form:

Form:	CALL QJOB (lun[,nHname[,flags[,parm[,ev]]]])
Where:	lun is an integer specifying the (virtual) logical unit number on which the job file resides n is the number of characters in name name of job file is a string of one to five ASCII characters (file extension is always JOB) flags is an integer containing the job flags parm is an integer containing the job parameters ev is the integer event variable
Example:	CALL QJOB (17,4HSCAN,2,31,IEV)

The CAL parameter block (CPB) for this directive consists of:

Word	Contents
0	CAL function code (33)
1	Event variable address
2	File name (first half)
3	File name (second half)
4	LUN
5	Job flags
6	Job parameters
7	Login device and unit
10	Login UFD

The file name and the login UFD are coded in .SIXBT format. The file extension of all job files is JOB. CPB words 7 and 10 are not used unless bit 7 (UFDFLG) of word 5 is set.

The contents of CPB word 4 (LUN) are:

Bit	Meaning
0-8	LUN containing the job file
9-17	Reserved for the LUN for the listing file

The job file JUN should be specified as a normal integer in the FORTRAN call. QJOB positions the LUN number correctly.

The contents of CPB word 5 (job flags) are:

Bit	Name	Meaning
0	DLTFLG	Delete job file after processing
1	OPRFLG	Operator required to run this job
2	FRCFLG	Force this job to run next
3	SEQFLG	Sequence this job (sequenced jobs are run in the order of submittal)
4		Reserved
5	HLDFLG	Hold this job until it is released by the operator
6	CCLFLG	Reserved for RSX CCL (concise command language)
7	UFDFLG	Login device, unit and UFD is specified in CPB words 7 and 10
8-17	TIMMSK	Job file time limit in minutes (zero implies that the default value is used)

UFDFLG must be clear (zero) in the FORTRAN call.

The contents of CPB word 6 (job parameters) are:

Bit	Name	Meaning
0-2	CLSMSK	Job priority class (0 to 7)
3-10		Reserved
11-17	MEMMSK	Minimum core needed to run the job (zero indicates 1K; all ones, 127, indicates 128K)

The contents of CPB word 7 (login device and unit) are:

Bit	Meaning
0-11	Login device, expressed in .SIXBT format
12-17	Login unit, expressed in binary

If CPB word 7 is zero, the system device is implied. If CPB word 10 (login UFD) is zero, SCR is implied.

The login device, unit and UFD in which the job will run is determined in the following way:

1. If UFDFLG (bit 7 of CPB word 5) is set, the login device, unit and UFD are specified by CPB words 7 and 10. If either CPB word is zero, appropriate system defaults are used.

2. If UFDFLG is reset (zero) and the job file resides on a disk, the device, unit and UFD in which the job file resides is used.

3. If UFDFLG is reset and the job file does not reside on a disk, the system defaults (system disk and SCR) are used.

If the directive is rejected, the event variable (if specified) is set to one of the following values to indicate rejection and the cause:

Event Variable	Meaning
-101	LUN out of range
-102	LUN not assigned
-105	Device unit number out of range
-106	Too many jobs queued
-777	Deque node is not available (POOL empty)

If the directive is accepted, the event variable (if specified) is set to #n, where n is the job file sequence number assigned to the job. The job file sequence number, used in conjunction with the date of submittal, constitutes a unique identifier for each batch job file.

This page intentionally left blank.

This page intentionally left blank.

2.13 QUEUE I/O: QUEUING REQUESTS FOR AN I/O DEVICE

The QUEUE I/O directive instructs RSX to place an I/O request for a particular device unit in a queue of requests for that unit. Entries in the queue are ordered according to task priority. Each I/O call generates a unique version of the QUEUE I/O directive. The user task specifies an operation (e.g., READ, WRITE, REWIND), a logical unit number (LUN) corresponding to a physical I/O device unit, and additional arguments specific to the operation.

There are two types of LUNs in RSX: system LUNs and virtual LUNs. A block of 25 virtual LUNs for each user is "mapped" to a block of system LUNs maintained by RSX. Each user is aware of only his own virtual LUNs. The system automatically relates virtual LUNs to system LUNs whenever a user requests a task to be run. Refer to Part VII of this manual for more information about LUNs. The virtual LUN included in the call identifies the physical device for which requests are queued. The user task can optionally include an event variable.

QUEUE I/O requests can be issued by both MACRO and FORTRAN programs. No comparable Monitor Console Routine call exists.

The generalized CAL parameter block (CPB) for all forms of the QUEUE I/O directive consists of the following:

Word	Contents
0	CAL function code (00) (bits 12 to 17); I/O function code (bits 3 to 11)
1	Event variable address
2	Logical unit number (LUN)
3	Unique to I/O call
4	Unique to I/O call
5	Unique to I/O call

If the QUEUE I/O directive is rejected by RSX, the event variable (if specified) is ordinarily set to one of the following values to indicate rejection and the cause:

Event Variable	Meaning
-101	LUN out of range
-102	LUN not assigned to a physical device
-103	Nonresident (not loaded) or noninitialized I/O device handler task
-777	Deque node (for PDVL) not available (POOL empty)

If the call is accepted, the event variable (if specified) is set to zero. A request node is formed and entered in the queue associated with the specified LUN at the priority of the task that issues the call. The handler task is "triggered" to signal that a request has been made. Then control is returned to the user task. The user task can subsequently test for completion of the operation by checking for a nonzero event variable.

Each request node has the following format:

Word	Contents
0	Forward linkage
1	Backward linkage
2	Task STL node address (zero if the directive is not issued at API level 7)
3	Task PBDL node address
4	Calling task priority in decimal range 1 to 512, but ABORT requests issued by IORD (I/O Rundown) set the priority to 0.
5	I/O function code (bits 9 to 17); logical unit number (bits 0 to 8)
6	Event variable address (zero if none specified)
7	CPB word 3 (unique to I/O call)
10	CPB word 4 (unique to I/O call)
11	CPB word 5 (unique to I/O call)

If the QUEUE I/O directive is issued from an interrupt service routine (which runs at API level 0), words 2 and 3 are zero and word 4 (priority) is +1.

Following is an example of a QUEUE I/O CPB:

```
CPB 2600     /CAL PARAMETER BLOCK TO READ (I/O FUNCTION
    EV       /CODE = 26;  CAL FUNCTION CODE = 00)
    4        /FROM LUN #4
    2        /IN IOPS ASCII (MODE 2).
    BUFFER   /TRANSFER TO CORE IS TO BEGIN AT
    100      /"BUFFER" AND CONTINUE FOR NO MORE THAN
             /100 (OCTAL) WORDS.
```

2.14 RAISEB: RAISING THE MEMORY-PROTECT BOUND

The RAISEB Directive instructs RSX to increase the Task size of the calling Task and, if the Task is built in USER-mode, to raise its memory-protect bound. Because a Task's I/O buffers are created from the unused space at the top of a Task's partition, Task size is not quite equivalent to partition size. RAISEB returns to the calling Task the highest usable Task address in the partition. This parameter is used by such system Tasks as the Assembler to create symbol table space as large as the partition allows. The RAISEB Directive is typically used after a Task has reserved I/O buffer space by means of the PREAL I/O request.

The actual raising of the memory-protect bound does not occur until the next Significant Event. Therefore it is recommended that a WAIT Directive follow the RAISEB Directive to provide the appropriate delay.

No FORTRAN call or standard system MACRO has been implemented for RAISEB.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (27)
1	Event Variable address

If a RAISEB Directive cannot be honored, the following Event Variable is returned to indicate rejection and its cause:

Event Variable	Meaning
-203	CAL not Task-issued

If the Directive is accepted, the Event Variable is set to the highest usable Task address (relative to the partition base for USER-mode Tasks).

Following is an example of a RAISEB CPB:

```
CPB 27   /RAISE BOUND AND RETURN
    EV   /HIGHEST USABLE ADDRESS
         /IN EV.
```

2.15 REQUEST: REQUESTING TASK EXECUTION

The REQUEST Directive instructs RSX to activate a Task and to REQUEST its execution at a specified software priority. Actual time of execution depends on partition availability and on Task priority. The Task cannot be brought into core from disk until its core partition is free. It cannot execute until all higher-priority active Tasks have relinquished control. The user can specify a new priority or can REQUEST execution at the Task's default priority (assigned during Task installation). He can optionally specify an Event Variable in the Directive.

Because a priority must be included in the FORTRAN call, priority 0 indicates the default priority. This is functionally equivalent to omitting the priority specification because Tasks run at API level 7, whose priorities range only from 1 to 512. A priority should be specified in the MACRO call, but 0 is used to indicate that the default priority is intended.

REQUESTs may be issued by MACRO and FORTRAN programs and by interrupt service routines, but a Task cannot REQUEST itself. The MACRO call has the following form:

Form:	REQUEST name[,p[,ev]]
Where:	name of Task REQUESTed is a string of one to six .SIXBT characters p is an integer priority in range 0-512 decimal ev is the Event Variable address
Examples:	REQUEST execution of SCAN at the default priority of 48: REQUEST SCAN,EV or REQUEST SCAN,0,EV or REQUEST SCAN,48,EV Priority redefined here at 20: REQUEST SCAN,20,EV Priority redefined here at 20 with no Event Variable: REQUEST SCAN,20

The FORTRAN call has the following form:

Form:	CALL REQST(nHname,p[,ev])
Where:	n is the number of characters in name name of Task REQUESTed is a string of one to five ASCII characters p is an integer priority in range 0-512 decimal ev is the integer Event Variable
Examples:	REQUEST execution of SCAN at the default priority of 48: CALL REQST (4HSCAN,0,IEV) or CALL REQST (4HSCCAN,48,IEV) Priority redefined in FORTRAN program: IP = 20 CALL REQST (4HSCAN,IP,IEV) REQUEST execution at SCAN's default priority with no Event Variable testing: CALL REQST (4HSCAN,0)

The REQUEST Directive can also be called from the Monitor Console Routine.

The CAL Parameter Block (CPB) for this Directive consists of the following:

Word	Contents
0	CAL function code (01)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)
4	Priority (0-512)

If REQUEST is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system
-202	Task is already or still active
-204	Task is disabled
-206	Illegal Task priority
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured
-777	Deque node (for ATL) not available (empty pool)

If the Directive is accepted, the Event Variable (if specified) is set to +1 and the REQUESTed Task is made active at priority p (or at its default priority if p is zero).

Following is an example of a REQUEST CPB:

```
CPB 01                 /CAL PARAMETER BLOCK.
    EV                 /REQUEST EXECUTION OF TASK
    .SIXBT "T1@@@@"    /"T1" NOW.  TASK IS TO RUN
    0                  /AT ITS DEFAULT PRIORITY.
```

2.15A REQUEST MAPPED: REQUESTING A TASK WITH MAPPED LUNS

The REQUEST MAPPED directive is nearly identical to the REQUEST directive, with two differences. First, the CAL code for REQUEST MAPPED (i.e., the first word of the CPB) is octal 36 instead of 1. Second, the REQUEST MAPPED CPB contains an additional word. The new CPB word (word 5) follows the word for task run priority (word 4) in the CPB. Word 5 of the REQUEST MAPPED CPB contains:

Bit	Contents
0-8	LUN offset (i.e., the octal number of the system LUN into which virtual LUN-2 is mapped)
9-14	MULTIACCESS user number
15-17	Unused

An example of the REQUEST MAPPED CPB is:

```
36                  /REQUEST TASK "T1" WITH
EV                  /MAPPED LUNS.  MAKE
.SIXBT "T1@@@@"     /REFERENCES TO LUN-2 GO
0                   /TO SYSTEM LUN-64
100000              /(OCTAL 100)
```

If the REQUEST MAPPED directive is rejected by RSX, the event variable (if specified) is set to one of the same values as described for the REQUEST directive, with the following additional possible value:

Event Variable	Meaning
-101	LUN out of range (indicates that the map requested does not fit in the real LUN space)

2.16 RESUME: RESUMING TASK EXECUTION

The RESUME Directive instructs RSX to RESUME execution of a suspended Task. All Tasks RESUME at the priority at which they were running when suspended. The user can optionally include an Event Variable.

The Directive can be issued by MACRO and FORTRAN programs and also by interrupt service routines. If RESUME is issued by a MACRO program and the suspended Task is in EXEC mode, execution of the Task can RESUME at any location within the partition. If the address is set to zero, the suspended Task RESUMEs at the location immediately following the SUSPEND Directive. USER-mode Tasks RESUMEd by MACRO programs restart at the next location. A FORTRAN call cannot include a resumption address, so all Tasks RESUMEd by FORTRAN programs restart at the next location.

The MACRO call has the following form:

Form:	RESUME name[,ra[,ev]]
Where:	name of Task RESUMEd is a string of one to six .SIXBT characters ra is the resumption address ev is the Event Variable address
Example:	RESUME TSKA,RSTRT,EVA

A case where a resumption address may be specified symbolically is in an interrupt service routine, assembled as part of the Task that it is going to RESUME at different locations in the Task depending on the nature of the interrupt.

The FORTRAN call has the following form:

Form:	CALL RESUME(nHname[,ev])
Where:	n is the number of characters in name name of Task RESUMEd is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL RESUME(4HTSKA,IEV)

The RESUME Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (07)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)
4	Resumption address

Bits 0-2 of the resumption address (word 4) are ignored.

If the RESUME Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-30	Specified address for USER-mode Task is outside the Task's partition
-202	Task not active
-205	Task not suspended

If the Directive is accepted, the Event Variable (if specified) is set to +1, and execution of the indicated Task is RESUMEd (contingent on its priority) at the appropriate address.

The user should be aware of certain limitations on the use of resumption addresses. If the issuing Task has been built in USER mode, it may not specify a resumption address for the Task to be RESUMEd. If the Task to be RESUMEd has been built in USER mode, any resumption address for this Task is relative to the partition base.

Following is an example of a RESUME CPB:

```
CPB 07                  /CAL PARAMETER BLOCK
    EV                  /TO RESUME EXECUTION OF
    .SIXBT "T8@@@@"     /TASK "T8" AT LOCATION
    ABC                 /"ABC".
```

2.17 RUN: ACTIVATING TASK EXECUTION

The RUN Directive instructs RSX to initiate the execution of a Task at a specified software priority and after the passage of some time interval, with an optional reschedule interval. Actual time of execution depends on Task priority and partition availability. This Directive performs the same function as SCHEDULE, except that the time of execution is expressed as time from now, not as absolute time of day. The user can request that the Task RUN at its default priority (assigned during Task installation) or can specify a new priority when the Directive is issued. An Event Variable can optionally be included.

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 Table 2-2 below. The initial activation time interval must be included in the command line, but the reschedule interval is optional. If the reschedule interval is omitted or has a value of 0, the Task is executed only once. If the user specifies a reschedule interval of 0, he must also specify a reschedule unit, but it does not matter which unit he chooses. The Clock Interrupt Service Routine performs Task activation and rescheduling.

Table 2-3
Reschedule Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	1	0-262143
Second	2	0-86400
Minute	3	0-1440
Hour	4	0-24

Because a priority must be included in the FORTRAN call, priority 0 indicates the default priority. This is functionally equivalent to omitting the priority specification because Tasks run at API level 7, whose priorities range only from 1 to 512. A priority should be specified in the MACRO call, but 0 is used to indicate that the default priority is intended.

RUN Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. A Task may RUN itself. The MACRO call has the following form:

Form:	RUN name,si,su[,ri,ru[,p[,ev]]]
Where:	name of Task initiated is a string of one to six .SIXBT characters si is an integer representing the number of units which must elapse before the Task is requested (range given in Table 2-2) su is 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-2) ri is an integer representing the number of units which must elapse before the Task is rescheduled (range given in Table 2-2) ru is 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-2) p is an integer priority in range 0-512 decimal ev is the Event Variable address
Examples:	RUN INITS at a new priority of 512, 5 seconds from now, and reschedule it every 10 minutes thereafter: RUN INITS,5,2,10,3,512,EV RUN INITS at its default priority with no Event Variable testing: RUN INITS,5,2,10,3 RUN INITS only once: RUN INITS,5,2,0,3 or RUN INITS,5,2

The FORTRAN call has the following form:

Form:	CALL RUN(nHname,t,p[,ev])
Where:	n is the number of characters in name name of Task initiated is a string of one to five ASCII characters t is an array consisting of four integer words; it represents the initial activation time and reschedule interval and units: t(1) = initial activation interval (range given in Table 2-2) t(2) = initial activation units; may be 1, 2, 3, or 4 (see Table 2-2) t(3) = reschedule interval (range given in Table 2-2) t(4) = reschedule units; may be 1, 2, 3, or 4 (see Table 2-2) p is an integer priority in range 0-512 decimal ev is the integer Event Variable
Example:	RUN INITS at a new priority of 512, 5 seconds from now, and reschedule it every 10 minutes thereafter; priority set in FORTRAN program: DIMENSION IT(4) • • • IT(1) = 5 IT(2) = 2 IT(3) = 10 IT(4) = 3 IP = 512 CALL RUN(5HINITS,IT,IP,IEV)

The RUN Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (03)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)
4	Initial activation interval (see Table 2-2)
5	Initial activation units (see Table 2-2)
6	Reschedule interval (see Table 2-2)
7	Reschedule units (see Table 2-2)
10	Priority (0-512)

If the RUN Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-104	Illegal activation of reschedule units code
-201	Task not in system
-203	Directive not Task-issued
-204	Task is disabled
-206	Illegal Task priority
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured
-777	Deque node (for Clock Queue) not available (empty pool)

If the Directive is accepted, the Event Variable (if specified) is set to +1, and a request to make the indicated Task active is placed in the Clock Queue to come due at the appropriate time.

Following are examples of a RUN CPB:

```
CPB 03                   /CAL PARAMETER BLOCK TO
    0                    /RUN TASK "T3" 90 SECONDS
    .SIXBT "T3@@@@"      /FROM NOW.  NO RE-SCHEDULING.
    .DEC                 /USE DEFAULT PRIORITY.
    90; 2                /NO EVENT VARIABLE SPECIFIED.
    0;0;0
    .OCT

CPB 03                   /CAL PARAMETER BLOCK TO
    EV                   /RUN TASK "T4" NOW, AND
    .SIXBT "T4@@@@"      /EVERY TWO MINUTES
    .DEC                 /FROM NOW ON.  TASK IS
    0; 3                 /TO RUN AT ITS DEFAULT
    2; 3                 /PRIORITY LEVEL.
    0
    .OCT
```

2.18 SCHEDULE: SCHEDULING TASK EXECUTION

The SCHEDULE Directive instructs RSX to SCHEDULE the execution of a Task at some future time of day and at a specified software priority, with the option to reSCHEDULE it at periodic intervals. Actual time of execution depends on Task priority and partition availability. The user can SCHEDULE execution at the Task's default priority (assigned during Task installation) or can specify a new priority when the Directive is issued. An Event Variable can optionally be included.

The SCHEDULE time is expressed as absolute time of day. The reSCHEDULE interval is expressed in 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. Table 2-1 illustrates permissible ranges for each interval unit. The SCHEDULE time must be included in the command line, but the reSCHEDULE interval is optional. If the reSCHEDULE interval is omitted or has a value of 0, the Task is executed only once. If the user specifies a reSCHEDULE interval of 0, he must also specify a reSCHEDULE unit, but it does not matter which unit he chooses. Ticks are intervals equal to the period of the real-time clock. The Clock Interrupt Service Routine performs Task activation and reSCHEDULing.

Table 2-4
Reschedule Interval Ranges

(UNIT SYMBOL)

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	1	0-262143
Second	2	0-86400
Minute	3	0-1440
Hour	4	0-24

Because a priority must be included in the FORTRAN call, priority 0 indicates the default priority. This is functionally equivalent to omitting the priority specification because Tasks run at API level 7, whose priorities range only from 1 to 512. A priority should be specified in the MACRO call, but 0 is used to indicate that the default priority is intended.

SCHEDULE Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. A Task may SCHEDULE itself. The MACRO call has the following form:

Form:	SCHEDULE name,hour,minute,second[,ri,ru[,p[,ev]]]
Where:	name of Task SCHEDULEd is a string of one to six .SIXBT 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 2-1) ru is 1, 2, 3, or 4, representing the relevant unit of time (see Table 2-1) P is an integer priority in range 0-512 decimal ev identifies the Event Variable address
Examples:	SCHEDULE execution of SCAN at 4:30 p.m. at a new priority of 200 and reSCHEDULE it every 5 minutes thereafter: SCHEDULE SCAN,16,30,0,5,3,200,EV SCHEDULE execution at SCAN's default priority with no Event Variable testing: SCHEDULE SCAN,16,30,0,5,3 SCHEDULE execution of ALPHA only once at its default priority at 7:15 a.m. with no Event Variable testing: SCHEDULE ALPHA,7,15,0

The FORTRAN call has the following form:

Form:	CALL SCHED(nHname,t,p[,ev])
Where:	n is the number of characters in name name of Task SCHEDULEd is a string of one to five ASCII characters t is an array consisting of five integer words; it represents the SCHEDULE time and reSCHEDULE interval and units: t(1) = SCHEDULE hour in range 0-23 decimal t(2) = SCHEDULE minute in range 0-59 decimal t(3) = SCHEDULE second in range 0-59 decimal t(4) = reSCHEDULE interval (range given in Table 2-1) t(5) = reSCHEDULE units; may be 1, 2, 3, or 4 (see Table 2-1) p is an integer priority in range 0-512 decimal ev is the integer Event Variable
Examples:	SCHEDULE execution of SCAN at 4:30 p.m. at a new priority of 200 and reSCHEDULE it every 5 minutes thereafter; priority set in FORTRAN program: DIMENSION IT(5) . . IT(1) = 16 IT(2) = 30 IT(3) = 0 IT(4) = 5 IT(5) = 3 IP = 200 IEV = 0 CALL SCHED(4HSCAN,IT,IP,IEV) or CALL SCHED(4HSCAN,IT,200,IEV) SCHEDULE execution at SCAN's default priority with no Event Variable testing (same DIMENSION and array assignments): . . IP = 0 CALL SCHED(4HSCAN,IT,IP) SCHEDULE execution only once at SCAN's default priority with no Event Variable testing (same DIMENSION and array assignments): . . IT(4) = 0 IP = 0 CALL SCHED(4HSCAN,IT,IP)

The SCHEDULE Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (02)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)
4	SCHEDULE hour (0-23)
5	SCHEDULE minute (0-59)
6	SCHEDULE second (0-59)
7	ReSCHEDULE interval (see Table 2-4)
10	ReSCHEDULE units (see Table 2-4)
11	Priority (0-512)

If the SCHEDULE Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-104	Illegal reschedule unit code
-201	Task not in system
-203	Directive not Task-issued
-204	Task is disabled
-206	Illegal Task priority
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured
-777	Deque node (for Clock Queue) not available (empty pool)

If the Directive is accepted, the Event Variable (if specified) is set to +1 and a request to make the indicated Task active is placed in the Clock Queue to come due at the appropriate time.

Following is an example of a SCHEDULE CPB:

```
CPB 02                  /CAL PARAMETER BLOCK TO
    EV                  /SCHEDULE TASK "T2" AT
    .SIXBT "T2@@@@"     /17:29:45 HOURS, AND
    .DEC                /EVERY FIVE MINUTES
    17; 29; 45          /THEREAFTER.  TASK IS
    5; 3                /TO RUN AT PRIORITY
    300                 /LEVEL 300.
    .OCT
```

2.19 SETJEA: INITIALIZING FLOATING-POINT EXIT REGISTER

The SETJEA Directive instructs RSX to initialize the Floating-Point Hardware JMS Exit Address (JEA) register for the issuing Task. This Directive is used by the FORTRAN library run-time subroutine that handles floating-point hardware errors. SETJEA Directives are ordinarily not issued from user-written code. Therefore no FORTRAN call or standard system MACRO has been implemented for SETJEA.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (17)
1	Event Variable address
2	Subroutine entry address to be stored in the JEA

If the SETJEA Directive cannot be honored, one of the following Event Variables is returned to indicate rejection and its cause:

Event Variable	Meaning
-1	No FP hardware on this machine
-30	JEA address for USER-mode Task is outside the Task's partition
-203	CAL not Task-issued

If the SETJEA Directive is accepted, the Event Variable is set to +1, the JEA is initialized, and the address value is saved in the Task's Partition Block Node so that the JEA can be restored when necessary.

Following is an example of a SETJEA CPB:

```
  CPB 17        /SET JEA TO TRANSFER TO
      EV        /ERROR SUBROUTINE WHEN
      ERRSUB    /FP ERROR OCCURS.
/
/NOTE -- ERROR SUBROUTINE HAS 4 ENTRY POINTS
/
ERRSUB          0
                JMP OVR    /GO TO OVERFLOW
                0
                JMP UND    /GO TO UNDERFLOW
                0
                JMP DIV    /GO TO DIVIDE
                0
                JMP TRAP   /GO TO MEMORY VIOLATION (USER
                           /MODE (MEMORY PROTECTION) NOT
                           /DISABLED)
```

2.20 SHARE: INVOKING CORE SHARING

The SHARE directive permits user mode tasks to invoke core memory sharing and initially select or change pointers to the area of memory associated with the task's External Shared Address Space (ESAS) or to deactivate memory sharing entirely. The user can optionally include an Event Variable in the Directive.

This Directive can be issued by MACRO and FORTRAN programs but not by Interrupt Service routines. Since memory sharing pertains only to user mode tasks, exec mode tasks can issue this Directive but it will have no effect on their operation. The user should be aware that a task can specify only one shared area of memory at a time, but it can subsequently switch shared areas by re-issuing the SHARE Directive with other parameters. Note that Cal Parameter Blocks, Event Variables, Control Tables, and I/O in general cannot be specified within or for a task's ESAS region. Should a task make such an attempt, a bad CAL may result or, in the case of an I/O operation, a negative event variable will typically be returned.

The FORTRAN call has the following form:

Form	CALL SHARE (ANAME,IOFF,LEN,IACC[IEV])
Where:	ANAME is a real or double integer array consisting of the name of a shareable partition or system common block. The name may contain 1 to 5 ASCII characters. IOFF is an integer constant or variable with the offset from the area base address where ESAS is to start. LEN is an integer constant or variable specifying a code indicating the length of ESAS. Either Ø,1,2,3, or 4 are legal values to indicate length of Ø (deactivates memory sharing) 256, 768, 384Ø, or 7936 words respectively. IACC is an integer constant or variable specifying the desired access to ESAS. Zero indicates the task desires read only privileges; one indicates a desire for read, write privileges. IEV is the Integer Event Variable
Example:	CALL SHARE(PART,Ø,2,Ø IEV)

The CPB for the SHARE Directive consists of the following:

Word	Contents
Ø	CAL Function Code (35)
1	Event Variable Address
2	First half of area name in .SIXBIT format
3	Second half of area name in .SIXBIT format
4	Offset from area base address where ESAS is to start
5	Length of ESAS and access type desired flag

Word 5 contains two fields for data. Bit Ø of this word is zero if the task desires read only privileges; Bit Ø is set if a read, write capability is desired. Bits 1-17 of word 5 can be set to Ø (to deactivate memory sharing), 4ØØ, 14ØØ, 74ØØ, or 174ØØ octal to indicate the size of ESAS.

If the SHARE directive is accepted, the Event Variable is set to +1 and memory sharing will be activated for the task.

If the SHARE directive cannot be honored, one of the following Event Variables is returned to indicate rejection and its causes.

-32	Partition or System Common Block not in system or does not permit memory sharing
-77	Access type desired not consistant with access type permitted.
-1Ø4	Offset not a multiple of 4ØØ octal or illegal length specified, or base of shared area plus size of ESAS plus offset out of bounds.
-2Ø3	Directive not task-issued.
-213	Partition or System Common currently being reconfigured.

The following is an example of the SHARE CPB.

```
CPB 35                /CAL PARAMETER BLOCK TO
     EV               /SHARE 14ØØ OCTAL
     .SIXBT "SYS"     /WORDS OF "SYSCOM" IN
     .SIXBT "COM"     /READ/WRITE FASHION
     1ØØØ             /OFFSET BY 1ØØØ OCTAL
     4Ø14ØØ           /WORDS FROM ITS BASE
```

2.21 SPY AND SPYREL: EXAMINING CORE LOCATIONS

The SPY Directive allows both EXEC and USER Tasks to examine or SPY core locations anywhere in memory. The user can optionally include an Event Variable in the Directive. This Directive can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. The FORTRAN call has the following form:

Form:	CALL SPY(address, value[,ev])
Where:	address is the octal location to be examined value is an integer variable which will contain the contents of the specified address on return ev is the integer Event Variable
Example:	CALL SPY(RSTRT,IVAL,IEV)

This Directive cannot be called from the Monitor Console Routine.

The CPB for it consists of the following:

Word	Contents
0	CAL function code (31)
1	Event Variable address
2	Address to be examined
3	Returned contents of address to be examined

If the Directive is rejected, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-104	Parameter error (e.g., invalid address)
-203	Directive not Task-issued

If the Directive is accepted, the Event Variable (if specified) is set to +1, and the contents of the specified address returned.

Following is an example of a SPY CPB:

```
CPB 31     /CAL PARAMETER BLOCK TO
    EV     /EXAMINE LOCATION ABC AND
    ABC    /STORE CONTENTS IN VAL.
    VAL
```

The SPYREL subroutine executes the SPY Directive, but allows a USER Task to examine a core location in an area of memory relative to a lower bound defined by the contents of SPY 1, location 321. The address to be examined is computed by adding the address included in the call to the contents of absolute location 321. Location 321 must have been set by an EXEC-mode Task. The sum of the two numbers is the address whose contents will be placed in the value variable specified in the SPYREL call. The user can optionally include an Event Variable in the Directive. If the SPY call executed by the subroutine returns a negative Event Variable, the subroutine returns control to the calling program. Thus it is best to specify the Event Variable.

SPYREL can be issued only by a FORTRAN program. The call has the following form:

Form:	CALL SPYREL(address,value[,ev])
Where:	address is the octal number to be added to the lower bound to compute the octal address to be examined value is an integer variable which will contain the contents of the specified address on return ev is integer Event Variable
Example:	CALL SPYREL(RSTRT,IVAL,IEV)

2.22 SPYSET: MODIFYING CORE LOCATIONS

The SPYSET Directive allows both EXEC and USER Tasks to modify core locations within bounds specified in the Directive. These locations may legally be outside the boundaries of the Task issuing the call. The lower bound is defined in absolute location 321 (known as SPY1) and the upper bound in location 322 (SPY2). Both SPY1 and SPY2 must be initialized by an EXEC-mode Task before a SPYSET is issued. The user can optionally include an Event Variable in the Directive. SPYSET Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines.

A FORTRAN program can issue a SPYSET Directive in the following way:

Form:	CALL SPYSET(address,value[,ev])
Where:	address is the octal number to be added to the lower bound to compute the address to be set value is an integer to replace the contents of the address ev is the integer Event Variable
Example:	CALL SPYSET(RSTRT,NEWVAL,IEV)

This Directive cannot be called from the Monitor Console Routine.

The CPB for it consists of the following:

Word	Contents
0	CAL function code (32)
1	Event Variable address
2	Address relative to lower bound
3	Value to replace contents of address

If the Directive is rejected, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-77	Violation of restricted usage of Directive
-104	Parameter error (e.g., invalid address)
-203	Directive not Task-issued

An Event Variable of -77 will be returned if either SPY1 or SPY2 has not been properly initialized. The Directive will also be rejected and an Event Variable of -104 returned if the address to be set is greater than the contents of SPY2. If the Directive is accepted, the Event Variable (if specified) is set to +1 and the contents of the specified address replaced.

Following is an example of a SPYSET CPB:

```
CPB 32      /CAL PARAMETER BLOCK TO
    EV      /CHANGE CONTENTS OF LOCATION
    XXY     /XXY TO 1001.
    1001
```

2.23 SUSPEND: SUSPENDING TASK EXECUTION

The SUSPEND Directive instructs RSX to SUSPEND execution of the issuing Task. The Task remains active in its core partition, but execution cannot proceed until the RSX Executive receives a RESUME Directive for this Task. An Event Variable is not included in the Directive.

Only the Task from which the SUSPEND Directive is issued can be SUSPENDed. SUSPEND Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines.

A CPB exists for SUSPEND, but a standard MACRO call has not been implemented. The SUSPEND Directive cannot be issued via the Monitor Console Routine, since a Task can only be suspended by itself and is protected against suspension by all other means, including the operator.

The CPB for this Directive consists simply of the following:

Word	Contents
0	CAL function code (06)

The CAL is issued as follows:

```
        CAL (6)         /SUSPEND TASK EXECUTION
```

A FORTRAN PAUSE statement can be used to SUSPEND execution of a FORTRAN Task until RESUMEd.

2.24 SYNC: SYNCHRONIZING TASK EXECUTION

The SYNC Directive instructs RSX to make a Task active at some future time and at an indicated software priority, with the option to reschedule this activation at periodic intervals. Actual time of execution depends on Task priority and partition availability. The user can request execution at the Task's default priority (assigned during Task installation) or can specify a new priority when the Directive is issued. An Event Variable can optionally be included.

This Directive performs the same function as SCHEDULE or 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. For example, a Task named TASK1 is scheduled by a program 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 time that must elapse can therefore vary depending on the time of day. The Clock Interrupt Service Routine performs Task activation and rescheduling.

The SYNC Directive is useful in SYNChronizing the execution of a number of Tasks, so that peak loading of the system can be avoided.

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 Table 2-3 below. The initial activation time interval must be included in the command line, but the reschedule interval is optional. If si, the initial activation interval, is 0, execution of the Task is initiated on the next occurrence of the schedule unit. If the reschedule interval is omitted or has a value of 0, the Task is executed only once. If the user specifies a reschedule interval of 0, he must also specify a reschedule unit, but it does not matter which unit he chooses.

Table 2-5
Reschedule Interval Ranges

Unit of Time	Interval Symbol	Legal Range, Decimal
Tick	1	0-262143
Second	2	0-86400
Minute	3	0-1440
Hour	4	0-24

Because a priority must be included in the FORTRAN call, priority 0 indicates the default priority. This is functionally equivalent to omitting the priority specification because Tasks run at API level 7, whose priorities range only from 1 to 512. A priority should be specified in the MACRO call, but 0 is used to indicate that the default priority is intended.

SYNC Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. A Task may SYNChronize itself. The MACRO call has the following form:

Form:	SYNC name,sz,si,su[,ri,ru[,p[,ev]]]
Where:	name of Task initiated is a string of one to six .SIXBT characters sz is 1, 2, 3, or 4 and represents the relevant SYNChronization unit (see Table 2-3) si is an integer representing the number of units which must elapse before the Task is initially activated (range given in Table 2-3) su is 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-3) ri is an integer representing the number of units which must elapse before the Task is rescheduled (range given in Table 2-3) ru is 1, 2, 3, or 4 and represents the relevant unit of time (see Table 2-3) p is an integer priority in range 0-512 decimal ev is the Event Variable address
Examples:	SYNChronize execution of FRED at a new priority of 20, 9 seconds after the next minute mark (i.e., it is now 14:27:47: run Task at 14:28:09) and reschedule it every 4 minutes thereafter: SYNC FRED,3,9,2,4,3,20,SYNEV SYNChronize execution of SCAN at a new priority of 21,10 seconds after the next minute next and reschedule it every hour thereafter: SYNC SCAN,3,10,2,1,4,21,SYNEV SYNChronize execution of FRED only once at a new priority of 21, 10 seconds after the next minute mark: SYNC FRED,3,10,2,2,21,SYNEV or SYNC FRED,3,10,2,0,4,21,SYNEV

The FORTRAN call has the following form:

Form:	CALL SYNC(nHname,t,p[,ev])
Where:	n is the number of characters in name name of Task initiated is a string of one to five ASCII characters t is an array consisting of five integer words; it represents the SYNChronization unit and the initial activation and reschedule intervals and units: t(1) = SYNChronization unit; may be 1, 2, 3, or 4 t(2) = initial activation interval t(3) = initial activation units; may be 1, 2, 3, or 4 t(4) = reschedule interval t(5) = reschedule units; may be 1, 2, 3, or 4 p is an integer priority in range 0-512 decimal ev is the integer Event Variable
Example:	SYNChronize execution of FRED at a new priority of 20, 9 seconds after the next minute mark, and reschedule it every 4 minutes thereafter. In the same program, SYNChronize execution of SCAN at 21, 10 seconds after the next minute mark, and reschedule it every hour thereafter:

```
   INTEGER FREDEV,SCANEV,SCANP,FREDP
   DIMENSION IT(5)
     .
     .
   IT(1) = 3
   IT(2) = 9
   IT(3) = 2
   IT(4) = 4
   IT(5) = 3
   FREDEV = 0
   FREDP = 20
   CALL SYNC(4HFRED,IT,FREDP,FREDEV)
     .
   SCANEV = 0
   SCANP = 21
   IT(2) = 10
   IT(4) = 1
   IT(5) = 4
   CALL SYNC(4HSCAN,IT,SCANP,SCANEV)
C  INSURE BOTH SYNC REQUESTS WERE ACCEPTED
   IF (SCANEV.OR.FREDEV.LT.0) GO TO 10
C  BOTH SYNC REQUESTS WERE ACCEPTED AT THIS POINT
     .
C  REPORT FAILURE OF SYNCS TO BE ACCEPTED TO
C  CONSOLE OPERATOR.
10 WRITE(3,11)
11 FORMAT(32H TASKS FRED OR SCAN NOT SYNC'ED.//)
   STOP
   END
```

The SYNC Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (14)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)
4	SYNC unit (see Table 2-3)
5	Initial activation interval (see Table 2-3)
6	Initial activation unit (see Table 2-3)
7	Reschedule interval (see Table 2-3)
10	Reschedule unit (see Table 2-3)
11	Priority (0-512)

If the Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-104	Illegal SYNC, activation of reschedule units code
-201	Task not in system
-203	Directive not Task-issued
-204	Task is disabled
-206	Illegal Task priority
-212	Partition for Task's STL node lost because of reconfiguration
-213	Partition assigned to Task currently being reconfigured
-777	Deque node (for Clock Queue) not available (empty pool)

If the Directive is accepted, the Event Variable (if specified) is set to +1, and a request to make the indicated Task active is placed in the Clock Queue to come due at the appropriate time.

Following are examples of a SYNC CPB:

```
CPB 14                      /CAL PARAMETER BLOCK TO
    EV                      /SYNC TASK "T5" 2.5 MINUTES
    .SIXBT "T5@@@@"         /AFTER THE NEXT HOUR, AND
    .DEC                    /EVERY 30 MINUTES THEREAFTER.
    4                       /TASK IS TO RUN AT PRIORITY
    150; 2                  /LEVEL 512.
    30; 3
    512
    .OCT

CPB 14                      /CAL PARAMETER BLOCK TO
    0                       /SYNC TASK "T6" ON THE NEXT
    .SIXBT "T6@@@@"         /MINUTE.  IT IS TO EXECUTE ONLY
    .DEC                    /ONCE AT ITS DEFAULT PRIORITY.
    3                       /NO EVENT VARIABLE IS SPECIFIED.
    0; 3
    0; 3
    0
    .OCT
```

2.25 TSKNAM: RETURNING THE NAME OF THE ISSUING TASK

The TSKNAM Directive instructs RSX to return the name of the issuing Task. One of the most common applications of this Directive is its use by the FORTRAN PAUSE Object-Time System routine (OTS) to print a Task name in the PAUSE message. TSKNAM Directives are ordinarily not issued by user-written code. Therefore no FORTRAN call or standard system MACRO has been implemented for TSKNAM.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (25)
1	Event Variable address
2	Task name buffer (first half)
3	Task name buffer (second half)

If a TSKNAM Directive cannot be honored, the following Event Variable is returned to indicate rejection and its cause:

Event Variable	Meaning
-203	CAL not Task-issued

If the TSKNAM Directive is accepted, the Task name will be stored in words 2 and 3 of the CPB and the Event Variable will be set as follows:

Bit of Event Variable	Meaning
0	Cleared upon acceptance of Directive
1	Set if task requires Floating Point hardware
2	Set if task runs in Bank mode
3	Set if task runs in User mode
4	Unused and set to zero
5-6	Correspond to XVM mode bits of task's STL node (always zeroed for Exec mode tasks)
7	Set if task has IOT permission (always zeroed for Exec mode tasks)
8-17	Task priority

Following is an example of the TSKNAM CPB:

```
          CPB 25   /REPLACE "XX" WITH
              EV   /THE NAME OF THIS
              XX   /TASK, I.E., WHO
              XX   /AM I?
```

2.26 UNFIX: FREEING A CORE PARTITION

The UNFIX Directive instructs RSX to nullify a FIX Directive, i.e., to 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. The user can optionally include an Event Variable in the Directive.

This Directive can be issued by MACRO and FORTRAN programs and also by interrupt service routines.

Following is the MACRO call:

Form:	UNFIX name[,ev]
Where:	name of Task UNFIXed is a string of one to six .SIXBT characters ev is the Event Variable address
Example:	UNFIX SCAN,EV

and the FORTRAN call:

Form:	CALL UNFIX(nHname[,ev])
Where:	n is the number of characters in name name of Task UNFIXed is a string of one to five ASCII characters ev is the integer Event Variable
Example:	CALL UNFIX(4HSCAN,IEV)

The UNFIX Directive can also be called from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (16)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the UNFIX Directive is rejected by RSX, the Event Variable (if specified) is set to one of the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system
-207	Task not fixed

If the UNFIX Directive is accepted, the Event Variable (if specified) is set to +1 and the Task is UNFIXed.

Following is an example of an UNFIX CPB:

```
CPB 16                  /CAL PARAMETER BLOCK
    EV                  /TO UNFIX TASK "T10".
    .SIXBT "T10@@@"
```

2.27 UNMARK: CANCELLING MARK-TIME REQUESTS

The UNMARK Directive instructs RSX to cancel all outstanding mark-time requests for the Task specified in the call. It exists specifically for the use of the I/O Rundown Task and is also employed by the Magtape I/O Handler Task. UNMARK Directives are ordinarily not issued by user-written code. Therefore no FORTRAN call or standard system MACRO has been implemented for UNMARK.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (23)
1	Event Variable address
2	Task name (first half)
3	Task name (second half)

If the Directive is rejected by RSX, the Event Variable is set to the following to indicate rejection and its cause:

Event Variable	Meaning
-201	Task not in system

If the Directive is accepted, the Event Variable is set to +1. All mark-time nodes in the Clock Queue for the specified Task are nullified, but not removed from this list. Note that nodes are not actually removed from the queue and returned to the "Pool of Empty Nodes" until the period of time specified in the appropriate MARK Directive has elapsed. It is therefore possible to exhaust the pool by issuing too many MARK Directives in a short period of time, even if MARKs are followed by UNMARKs.

Following is an example of an UNMARK CPB:

```
CPB 23                  /CAL PARAMETER BLOCK
    EV                  /TO UNMARK TASK "T13".
    .SIXBT "T13@@@"
```

2.28 WAIT: WAITING FOR THE NEXT SIGNIFICANT EVENT

The WAIT Directive instructs RSX to suspend execution of the issuing Task until the next Significant Event. Until such time, the Task remains in its core partition and on the Active Task List, but is dormant. Lower-priority Tasks are allowed to run. When the next Significant Event occurs, the issuing Task (contingent upon priority) resumes at the location immediately following the WAIT CAL instruction. No Event Variable may be included in the Directive.

WAIT Directives can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. It is the responsibility of the Task that issues the Directive to determine the meaning of the Significant Event that causes its resumption. A CPB exists for WAIT, but a standard MACRO call has not been implemented. The CAL is issued as follows:

```
          CAL (5)          /WAIT FOR NEXT SIGNIFICANT EVENT.
```

A FORTRAN PAUSE statement can be used to suspend execution of a FORTRAN Task until a test indicates that the next Significant Event has occurred.

This Directive cannot be called from the Monitor Console Routine. The CPB for it consists simply of the following:

Word	Contents
0	CAL function code (05)

2.29 WAITFOR: WAITING FOR AN EVENT VARIABLE TO BE SET

The WAITFOR Directive instructs RSX to examine a specified Event Variable; if the Event Variable is zero, WAITFOR suspends execution of the Task that issues the Directive until the Event Variable is set to a nonzero value. RSX examines the Event Variable every time a Significant Event occurs and a high priority Task is unable to execute. As soon as a nonzero value is detected, the suspended Task is resumed at the priority at which it was previously running. The Task is resumed at the instruction immediately following the location of the WAITFOR Directive.

This Directive can be issued by MACRO and FORTRAN programs, but not by interrupt service routines. WAITFOR Directives affect only the Tasks from which they are issued. Following is the MACRO call:

Form:	WAITFOR ev
Where:	ev is the Event Variable address
Example:	In this example, the MARK Directive sets Event Variable MRKEV to zero for 5 minutes and then resets it to a nonzero value. WAITFOR detects that the Event Variable is zero and suspends Task execution until it is reset to a nonzero value (i.e., in 5 minutes). Then the Task is resumed at the instruction immediately following the WAITFOR: MARK 5,3,MRKEV WAITFOR MRKEV

and the FORTRAN call:

Form:	CALL WAITFR(ev)
Where:	ev is the integer Event Variable

Example:	See introduction to MACRO example above:
	`   DIMENSION IT(2)`
	`      .`
	`      .`
	`      .`
	`   IT(1) = 5`
	`   IT(2) = 3`
	`C  IEV WILL AUTOMATICALLY BE CLEARED BY MARK`
	`   CALL MARK (IT,IEV)`
	`   CALL WAITFR(IEV)`

The WAITFOR Directive cannot be issued from the Monitor Console Routine.

The CPB for this Directive consists of the following:

Word	Contents
0	CAL function code (20)
1	Event Variable address

The subroutine which follows illustrates the interaction of MARK and WAITFOR CPBs:

```
DELAY 0             /SUBROUTINE TO SUSPEND EXECUTION
      DAC MARK+2    /FOR THE NUMBER OF SECONDS
      CAL MARK      /INDICATED BY THE CONTENTS OF AC.
      CAL WAITF
      JMP* DELAY
/
MARK 13             /CAL PARAMETER BLOCK TO CLEAR "EV"
      EV            /THEN SET IT AND DECLARE A
      XX            /SIGNIFICANT EVENT XX SECONDS FROM NOW.
      2
/
WAITF 20            /CAL PARAMETER BLOCK TO SUSPEND
      EV            /EXECUTION UNTIL "EV" IS SET (WAITFOR).
/
EV    0             /EVENT VARIABLE.
```

2.30 XFRCMD: TRANSFERRING TDV COMMAND LINE

The XFRCMD directive transfers the IOPS ASCII command line supplied by the user and read by the MULTIACESS Monitor to a specified line buffer in the partition of the issuing task. XFRCMD is used by all TDV function tasks. No FORTRAN call or standard system MACRO has been implemented for XFRCMD.

The CPB for XFRCMD consists of the following:

Word	Contents
0	CAL function code (37)
1	Event variable address
2	TDV task buffer address
3	TDV task buffer size (octal)

If the XFRCMD directive cannot be executed, one of the following event variables is returned to indicate rejection and the cause:

Event Variable	Meaning
-16	TDV task buffer too small
-77	Violation of restricted use of XFRCMD; the issuing task is not a TDV function
-203	CAL not task-issued

If the XFRCMD directive is accepted, the command line is transferred and the event variable is set to +1.

Following is an example of a XFRCMD CPB:

```
CPB 37    /TRANSFER THE COMMAND LINE
    EV    /TO "BUF".  THE BUFFER
    BUF   /SIZE IS 34 (OCTAL) WORDS.
    34
```

CHAPTER 3

SYSTEM MACROS

System Macros are similar to System Directives but are characterized by the following:

- System Macros are not implemented as CAL instructions; they are direct MACRO subroutine calls and, as such, may be executed only by Tasks built in EXEC mode.
- System Macros do not return Event Variable information.

INTENTRY and INTEXIT are not absolutely required. They are provided as a coding convenience and are recommended only when it is necessary to save and restore many active system registers.

3.1 INTENTRY: ENTERING REGISTER SAVE ROUTINE

The INTENTRY System Macro instructs RSX to enter the Executive's Register Save Routine. This routine obtains the current contents of all active system registers (e.g., AC, index and limit registers, autoincrement registers) and deposits the contents in a save area. This area is created by the MACRO Assembler during expansion of the INTENTRY System Macro.

INTENTRY System Macros may be issued only from MACRO interrupt service routines. The call itself must be the first instruction in this routine. Following is the MACRO call:

Form:	INTENTRY entry
Where:	entry is the octal entry address (connect location) of the interrupt service routine. The user must not supply the address tag entry because it is part of the code generated by this macro definition.
Example:	INTENTRY CL

Registers saved by INTENTRY are restored by execution of INTEXIT.

3.2 INTEXIT: ENTERING REGISTER RESTORE ROUTINE

The INTEXIT System Macro instructs RSX to enter the Executive's Register Restore Routine. This routine restores all active registers saved by INTENTRY, debreaks, and returns to the interrupted Task. INTEXIT System Macros can be issued only from MACRO interrupt service routines. Following is the MACRO call:

Form:	INTEXIT entry
Where:	entry is the octal entry address (connect location) of the interrupt service routine

Example:	Interrupt service routine ADINT (for A/D Converter) uses INTENTRY and INTEXIT:
	`INTENTRY ADINT    /MUST BE PLACED AT THE ENTRANCE`
	`  .               /  TO THE INTERRUPT ROUTINE.`
	`  .`
	`  .               /SECTION TO SERVICE INTERRUPT.`
	`INTEXIT ADINT     /RESTORE REGISTERS, DEBREAK,`
	`                  /  AND RETURN TO INTERRUPTED`
	`                  /  TASK.`

CHAPTER 4

EVENT VARIABLES

Event Variables are software flags set by RSX for system or other Tasks, and are used to indicate the success or failure of the Task's operations. Most System Directives provide for an Event Variable in the MACRO or FORTRAN call. With few exceptions, inclusion of this variable is optional. If the user does specify an Event Variable, a code indicating the status of the Directive is normally returned to the issuing Task after the request has been processed. This specified code is in the following ranges:

Code	Meaning
+n	Directive accepted; n is almost always 1
0	Directive pending
-n	Directive rejected: n is a number indicating why rejection occurred

The second CPB word for all Directives except EXIT, WAIT, and SUSPEND contains the address of the Event Variable to be set. If the contents of this word remain zero, this indicates that the user Task has specified no Event Variable to be set by the Directive within it or that, for some reason, the Directive request is still pending.

If the user Task does specify an Event Variable, it is initially set to zero to indicate that the Directive has not yet been processed. Once the Directive has been accepted or rejected, the Event Variable is set with a nonzero value to indicate either acceptance or rejection for a particular reason. In the sections above, Event Variables have been described for System Directives to which they apply. Some Event Variables are specific to a particular Directive; others are common to two or more Directives. Table 4-2 provides a summary of all negative Event Variables which may be returned to RSX System Directives, as well as possible reasons for failure, and Directives to which particular Event Variables are returned. Directives are abbreviated as follows:

Table 4-1
System Directive Abbreviations

Abbreviation	Directive	Function
IO	QUEUE I/O	Queue requests for I/O unit
RQ	REQUEST	Request task execution
SC	SCHEDULE	Schedule task execution
RN	RUN	Activate task execution
SY	SYNC	Synchronize task execution
CN	CANCEL	Cancel requests for a task
RS	RESUME	Resume task execution
MT	MARK	Set event variable in future
UM	UNMARK	Cancel mark-time requests
FX	FIX	Fix a task in core
UF	UNFIX	Free a core partition
DA	DISABLE	Disable a task
EA	ENABLE	Reenable a task
CI	CONNECT	Connect to interrupt line
DI	DISCONNECT	Disconnect from interrupt line
DT	DATE	Retrieve time and date
SJ	SETJEA	Initialize floating-point exit register
TN	TSKNAM	Return name of issuing task
PI	PARINF	Return partition address and size
RB	RAISEB	Raise memory-protect bound
TC	XFRCMD	Transfer TDV command line
SP	SPY	Examine core locations
SS	SPYSET	Modify core locations
QU	QJOB	Queue a batch job
SH	SHARE	Invoke memory sharing
EX	EXECUTE	Request task execution from user disk
RM	REQUEST MAPPED	Request a task with mapped LUNs

Table 4-2
Returned Event Variables

Event Variable	Directive	Reason
-1	SJ	No floating-point hardware on this machine
-2	IO	I/O request aborted
-16	TC	Output word-pair count or input buffer-size error
-26	CI,DI	Illegal function for a user-mode task
-30	SJ,RS	Address for user-mode task is outside the task partition
-32	SH	Nonexistant system COMMON block or partition, or core sharing not permitted
-77	TC,SS,SH,RM	Violation of restricted usage of directive
-101	IO,RM	LUN out of range
-102	IO	LUN not assigned to a physical device
-103	IO	Nonresident or noninitialized I/O device handler task
-104	SP,SS,SH, SY,SC,RN	Parameter error (in CPB or control table)
-201	RQ,SC,RN, SY,CN,UM, FX,UF,DA, EA,EX,RM	Task not in system
-202	RQ,FX,EX RS,RM	Task is active Task is inactive
-203	SC,RN,SY, MT,SJ,PI, TC,TN,RB, SP,SS,SH,QU	Directive not task-issued
-204	RQ,SC,RN, SY,FX,EX,RM	Task is disabled
-205	RS	Task not suspended
-206	RQ,SC,RN, SY,EX,RM	Illegal task priority
-207	FX UF	Task already fixed Task not fixed

(Continued on next page)

Table 4-2 (Cont.)
Returned Event Variables

Event Variable	Directive	Reason
-210	FX	Partition occupied
-211	PI	Partition not in system
-212	RQ,SC,RN,SY, FX,EA,EX,RM	Partition for task STL node lost because of reconfiguration
-213	RQ,SC,RN,SY, FX,EA,EX,SH,RM	Partition or system COMMON block currently being reconfigured
-301	CI,DI	Line number rejected
-302	CI DI	Line is connected Line is not connected
-401	EX	Illegal LUN
-402	EX	HINF error
-403	EX	Illegal device
-404	EX	Disk dismounted
-405	EX	GET error
-406	EX	File not created
-407	EX	File not found
-410	EX	ALLOCATE error
-411	EX	PUT error
-412	EX	Nonunique alias name
-777	Most modules	Deque node unavailable (empty pool)

APPENDIX A

THE MACRO DEFINITIONS FILE

All system MACROs implemented in the RSX system are defined in one of the files supplied as part of RSX and known as the MACRO Definitions File. This appendix contains definitions for all currently implemented standard system MACROs except those used exclusively for I/O.

The following listing summarizes abbreviations used in the MACRO Definitions File and appears at the beginning of that file.

```
/ EDIT #13
/
/ COPYRIGHT 1971,1972,1973 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
/
/ RSX-15 MACRO DEFINITIONS      14 JUN 73   H. KREJCI
/                                           M. HEBENSTREIT
/
/ ABREVIATIONS -- UNLESS OTHERWISE SPECIFIED, ALL PARAMETERS
/                 EXCEPT ADDRESSES ARE GIVEN IN DECIMAL.
/
/         BUFF      CORE BUFFER ADDRESS
/         CL        INTERRUPT CONNECT LOCATION
/         CTB       CONTROL TABLE ADDRESS
/         EV        EVENT VARIABLE ADDRESS
/         FLNAM     FILE NAME (1-6 CHARACTERS)
/         LN        INTERRUPT LINE NUMBER (OCTAL)
/         LUN       LOGICAL UNIT NUMBER
/         MI        MARK TIME INTERVAL (A TICK THRU A DAY)
/         MODE      DATA MODE INDICATOR
/         MU        MARK TIME UNITS
/         RA        RESUMPTION ADDRESS
/         RI        RESCHEDULE INTERVAL (0-1 DAY, WHERE 0 IMPLIES
                    NO RESCHEDULING)
/         RP        RUN PRIORITY (0-512, WHERE 0 IMPLIES DEFAULT
                    PRIORITY)
/         RU        RESCHEDULE UNITS (H,M,S,T)
/         SD        SCHEDULE DELTA (A TICK THRU A DAY)
/         SH        SCHEDULE HOURS (0-23)
/         SIZE      CORE BUFFER SIZE (OCTAL)
/         SM        SCHEDULE MINUTES (0-59)
/         SS        SCHEDULE SECONDS (0-59)
/         SU        SCHEDULE DELTA UNITS (H,M,S,T)
/         SZ        SYNCHRONIZATION UNIT (H,M,S,T)
/         TASNAM    TASK NAME (1-6 CHARACTERS)
/         EXT       FILE NAME EXTENSION (1-3 CHARACTERS)
/         UNIT      UNIT NUMBER OF DISK
/         TYPE      DISK TYPE: 2 -- RF, 3 -- RP, 24 -- RK, 0 -- SYSTEM
 DISK
/
H=4       /HOURS INDICATOR
M=3       /MINUTES INDICATOR
S=2       /SECONDS INDICATOR
T=1       /TICKS INDICATOR
/
.INH=705522 /INTERRUPT INHIBIT IOT
.ENG=705521 /INTERRUPT ENABLE IOT
/
SAVE=131 /SAVE ENTRY POINT (IN SCOM)
REST=134 /RESTORE ENTRY POINT (IN SCOM)
```

A.1 REQUEST: REQUESTING TASK EXECUTION

```
          .DEFIN REQUEST,TN,RP,EV
          CAL .+2
          JMP .+6
          01
          EV+0
          ..=.; .SIXBT "TN"
          0; .LOC ..+2
          .DEC
          RP+0
          .ENDM
```

A.2 SCHEDULE: SCHEDULING TASK EXECUTION

```
          .DEFIN SCHEDULE,TN,SH,SM,SS,RI,RU,RP,EV
          CAL .+2
          JMP .+13
          02
          EV+0
          ..=.; .SIXBT "TN"
          0; .LOC ..+2
          .DEC
          SH; SM; SS
          RI+0
          RU+0
          RP+0
          .ENDM
```

A.3 RUN: ACTIVATING TASK EXECUTION

```
          .DEFIN RUN,TN,SD,SU,RI,RU,RP,EV
          CAL .+2
          JMP .+12
          03
          EV+0
          ..=.; .SIXBT "TN"
          0; .LOC ..+2
```

```
        .DEC
        SD; SU
        RI+0
        RU+0
        RP+0
        .ENDM
```

A.4 SYNC: SYNCHRONIZING TASK EXECUTION

```
        .DEFIN SYNC,TN,SZ,SD,SU,RI,RU,RP,EV
        CAL .+2
        JMP .+13
        14
        EV+0
        ..=.; .SIXBT "TN"
        0; .LOC ..+2
        .DEC
        SZ; SD; SU
        RI+0
        RU+0
        RP+0
        .ENDM
```

A.5 CANCEL: CANCELLING REQUESTS FOR A TASK

```
        .DEFIN CANCEL,TN,EV
        CAL .+2
        JMP .+5
        04
        EV+0
        ..=.; .SIXBT "TN"
        0; .LOC ..+2
        .ENDM
```

A.6 SUSPEND: SUSPENDING TASK EXECUTION

```
        .DEFIN SUSPEND
        CAL (6)
        .ENDM
```

A.7 RESUME: RESUMING TASK EXECUTION

```
        .DEFIN RESUME,TN,RA,EV
        CAL .+2
        JMP .+6
        07
        EV+0
        ..=.; .SIXBT "TN"
        0; .LOC ..+2
        RA+0
        .ENDM
```

A.8 MARK: SETTING AN EVENT VARIABLE IN THE FUTURE

```
.DEFIN MARK,MI,MU,EV
CAL .+2
JMP .+5
13
EV
.DEC; MI; MU
.ENDM
```

A.9 WAITFOR: WAITING FOR AN EVENT VARIABLE TO BE SET

```
.DEFIN WAITFOR,EV
CAL .+2
JMP .+3
20
EV
.ENDM
```

A.10 WAIT: WAITING FOR THE NEXT SIGNIFICANT EVENT

```
.DEFIN WAIT
CAL (5)
.ENDM
```

A.11 EXIT: TERMINATING EXECUTION OF A TASK

```
.DEFIN EXIT
CAL (10)
.ENDM
```

A.12 CONNECT: CONNECTING TO AN INTERRUPT LINE

```
.DEFIN CONNECT,LN,CL,EV
CAL .+2
JMP .+5
11
EV+0
LN
CL
.ENDM
```

A.13 DISCONNECT: DISCONNECTING FROM AN INTERRUPT LINE

```
.DEFIN DISCONNECT,LN,CL,EV
CAL .+2
JMP .+5
12
EV+0
LN
CL
.ENDM
```

A.14 DISABLE: DISABLING A TASK

```
.DEFIN DISABLE,TN,EV
CAL .+2
JMP +5
21
EV+0
..=.; .SIXBT "TN"
0; .LOC ..+2
.ENDM
```

A.15 ENABLE: REENABLING A TASK

```
.DEFIN ENABLE,TN,EV
CAL .+2
JMP .+5
22
EV+0
..=.; .SIXBT "TN"
0; .LOC ..+2
.ENDM
```

A.16 FIX: FIXING A TASK IN CORE

```
.DEFIN FIX,TN,EV
CAL .+2
JMP .+5
15
EV+0
..=.; .SIXBT "TN"
0; .LOC ..+2
.ENDM
```

A.17 UNFIX: FREEING A CORE PARTITION

```
.DEFIN UNFIX,TN,EV
CAL .+2
JMP .+5
16
```

```
    EV+0
    ..=.; .SIXBT "TN"
    0; .LOC ..+2
    .ENDM
```

A.18 DATE: RETRIEVING TIME AND DATE

```
    .DEFIN DATE,MON,DAY,YEAR,HRS,MIN,SEC
    CAL .+2
    JMP .+11
    24
    0
MO  0
DA  0
YR  0
HH  0
MM  0
SS  0
    LAC MO
    DAC MON
    LAC DA
    DAC DAY
    LAC YR
    DAC YEAR
    LAC HH
    DAC HRS
    LAC MM
    DAC MIN
    LAC SS
    DAC SEC
    .ENDM
```

A.19 INTENTRY: ENTERING REGISTER SAVE ROUTINE

```
    .DEFIN INTENTRY,CL
CL  0
    DBA
    JMS* (SAVE)
    .REPT 24
    0
    .ENDM
```

A.20 INTEXIT: ENTERING REGISTER RESTORE ROUTINE

```
    .DEFIN INTEXIT,CL
    LAC (CL)
    JMP* (REST)
    .ENDM
```

APPENDIX B

REGISTERS SAVED DURING SAVE AND RESTORE OPERATIONS

Whenever the Executive switches control from one task to another, the registers listed below are saved for the outgoing task and restored for the incoming task if both tasks are in exec mode. The reentrant registers (R1 to R6) and the autoincrement registers (X10 to X17) are not saved or restored for user-mode tasks. Interrupt service routines can use the INTENTRY and INTEXIT system macros to do the same thing, except that the MM register, XM clock values and the floating-point hardware registers are not saved or restored.

The registers saved and restored are:

AC buffer
XR buffer
LR buffer
MQ buffer
SC buffer
R1 buffer
R2 buffer
R3 buffer
R4 buffer
R5 buffer
R6 buffer
X10 buffer
X11 buffer
X12 buffer
X13 buffer
X14 buffer
X15 buffer
X16 buffer
X17 buffer
L20 buffer
EPA buffer
FMA1 buffer
FMA2 buffer
FMQ1 buffer
FMQ2 buffer
JEA buffer
MM register buffer
XM clock overflows
XM clock ticks (above overflows)

APPENDIX C

.SIXBT CHARACTER SET

.SIXBT notation is employed to identify Task names in most System Directives described in this manual. .SIXBT denotes 6-bit ASCII characters, formed by truncating the leftmost bit of the corresponding 7-bit character. The following table supplies legal characters and codes for .SIXBT notation.

Printing Character	7-bit ASCII	.SIXBT	Printing Character	7-bit ASCII	.SIXBT
@	100	00	Form Feed	014	
A	101	01	Carriage Return	015	
B	102	02	Rubout	177	
C	103	03	(Space)	040	40
D	104	04	!	041	41
E	105	05	"	042	42
F	106	06	#	043	43
G	107	07	$	044	44
H	110	10	%	045	45
I	111	11	&	046	46
J	112	12	'	047	47
K	113	13	(	050	50
L	114	14	)	051	51
M	115	15	*	052	52
N	116	16	+	053	53
O	117	17	,	054	54
P	120	20	-	055	55
Q	121	21	.	056	56
R	122	22	/	057	57
S	123	23	0	060	60
T	124	24	1	061	61
U	125	25	2	063	62
V	126	26	3	063	63
W	127	27	4	064	64
X	130	30	5	065	65
Y	131	31	6	066	66
Z	132	32	7	067	67
[	133	33	8	070	70
\	134	34	9	071	71
]	135	35	:	072	72
↑	136	36	;	073	73
←	137	37	<	074	74
Null	000		=	075	75
Horizontal Tab	011		>	076	76
Line Feed	012		?	077	77
Vertical Tab	013				

INDEX