Chapter 5

Programmed Requests

This chapter describes program requests and lists EMT codes associated with specific programmed requests. For detailed information on programmed requests, see the *RT–11 System Macro Library Manual*.

EMT Codes

Table 5–1: EMT Codes 000–377

EMT Number	Interpretation
000-017	.DELETE V1
020-037	.LOOKUP V1
040-057	.ENTER V1
060-077	(.TRPSET) V1
100-117	.RENAME V1
120-137	.SAVESTATUS V1
140-157	.REOPEN V1
160-177	.CLOSE V1
200-217	.READ* V1
220-237	.WRIT* V1
240-257	.WAIT V1
260-277	(.CHCOPY) V1
300-317	(.DEVICE) V1
320-337	(.CDFN) V1
340	.TTYIN .TTINR
341	.TTYOUT .TTOUR
342	.DSTATUS
343	.FETCH addr >476
343	.RELEASE addr=0
344	.CSIGEN
345	.CSISPC .GTLIN
346	.LOCK
347	.UNLOCK
350	.EXIT
351	.PRINT

Table 5–1 (Cont.): EMT Codes 000–377

EMT Number	Interpretation
352	.SRESET
353	.QSET
354	.SETTOP
355	.RCTRLO
356	(.ASTX) Internal
357	.HRESET
360-366	Error Reporting Internal
367-372	Reserved
373	.CALLK
374	R0 contains argument
375	R0 points to arguments
376	Error Reporting Internal
377	Ignored (NOP)

Table 5–2: EMT 374 Subcodes

Subcodes		Interpretation
000	chn	.WAIT
001	000	.SPND
002	000	.RSUM
003	chn	.PURGE
004	000	.SERR
005	000	.HERR
006	chn	.CLOSE
007	000	.TLOCK
010	000	.CHAIN
011	000	.MWAIT
012	000	.DATE

Table 5–2 (Cont.): EMT 374 Subcodes

Subcodes		Interpretation
013	chn	.ABTIO
014-377		Reserved

Table 5–3: EMT 375 Subcodes

Subcodes		Interpretation
000	chn	.DELETE
001	chn	.LOOKUP
002	chn	.ENTER
003	000	.TRPSET
004	chn	.RENAME
005	chn	.SAVESTATUS
006	chn	.REOPEN
007	chn	(.CLOSE)
010	chn	.READ*
011	chn	.WRIT*
012	chn	(.WAIT)
013	chn	.CHCOPY
014	000	.DEVICE non-linking
014	001	.DEVICE linking
015	000	.CDFN
020	000	.GTJB V3
020	001	.GTJB V4+
021	000	.GTIM
022	000	.MRKT
023	000	.CMKT
024	000	.TWAIT
025	000	.SDAT*

Table 5–3 (Cont.): EMT 375 Subcodes

Subcodes		Interpretation
026	000	.RCVD*
027	000	.CSTATUS
030	000	.SFPA
031	000	.PROTECT
031	001	.UNPROTECT
032	chn	.SPFUN
033	000	.CNTXSW
034	000	.GVAL
034	001	.PEEK
034	002	.PVAL mov
034	003	.POKE mov
034	004	.PVAL bic
034	005	.POKE bic
034	006	.PVAL bis
034	007	.POKE bis
035	000	.SCCA local
035	001	.SCCA global
036	000	.CRRG
036	001	.ELRG
036	002	.CRAW
036	003	.ELAW
036	004	.MAP
036	005	.UNMAP
036	006	.GMCX
037	000	.MTSET
037	001	.MTGET
037	002	.MTIN
037	003	.MTOUT

Table 5–3 (Cont.): EMT 375 Subcodes

Subcodes		Interpretation
037	004	.MTRCTO
037	005	.MTATCH
037	006	.MTDTCH
037	007	.MTPRNT
037	010	.MTSTAT
040	000	.SDTTM
041	000	.SPCPS
042	000	.SFDAT
043	000	.FPROT
044	chn	.GFINF
044	chn	.GFSTA
044	chn	.GFDAT
044	chn	.SFINF
044	chn	.SFSTA
045	chn	.CLOSZ
046	000	.CMAP
046	000	.GCMAP 2(R0)=0
046	000	.MSDS bit 4000 2(R0) on
047-377		Reserved

Notes
chn = channel number
(.xxx) = No Macro .xxx defined to gen this version of the request.
Internal = reserved for internal use by monitor

.ABTIO

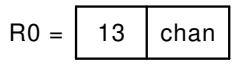
EMT 374, Code 13

The .ABTIO programmed request allows a job to abort all outstanding I/O operations on a channel without terminating the program.

.ABTIO chan

where:

chan is the channel number on which to abort I/O



Errors:
None.

.ADDR

Macro Expansion

The .ADDR macro computes the specified address in a position-independent manner.

.ADDR addr,reg,push

where:

addr is the label of the address to compute, expressed as an immediate value with a number sign (#) before the label.

reg is where to store the computed address, expressed as Rn or @Rn. To store the address on the stack, use @SP or -(SP). An @SP stores the address in the stack's current top, while -(SP) pushes the address onto a new location which becomes the top of the stack. The following register references are valid:

R0 — R5
@R0 — @R5
@SP — -(SP)

push determines what to do with the original contents of the register. If you omit push, the computed address overwrites the register contents. If you use ADD for the push argument, the computed address is added to the original contents of the register. If you use PUSH for the push argument, the register's previous contents are pushed onto the stack before the computed address is stored in the register.
If you use -(SP) for the argument *reg*, you may omit the push argument, since PUSH is automatically used.

.ASSUME

Macro Expansion

The .ASSUME macro tests the validity for a condition you specify. If the test is false, MACRO generates an assembly error and prints a descriptive message.

.ASSUME a rel c [message = <text>]

where:

a is an expression.

rel is the relationship between a and c you want to test. There can be six values for *rel*: {eq}, {ne}, {gt}, {et}, {ge}, and {le}.

c is an expression.

text is the message you want MACRO to print if the condition you specified in the relationship between *a* and *c* is false. To specify your own error message, start the message with a semicolon (;), or start with a valid assembly expression followed by a semicolon (;) and the message. If you omit the message argument, the error message *a rel c is not true* displays; the expressions you used appear in the message in place of *a* and *c*.

.BR

Macro Expansion

The .BR macro warns you during assembly time if code that belongs together is separated. When you invoke the .BR macro, you specify an address as an argument. .BR checks that the next address equals the address you specified in the .BR macro. If it does not, MACRO prints the error message: *ERROR; ?SYSMAC-E-Not at location addr*. The location you specified in the .BR macro appears in place of *addr* in the message. If you specify a symbol as an argument and the symbol is not defined in the current assembly, you will get an error message: *ERROR; ?SYSMAC-E-addr is not defined*. At assembly time, both .ASSUME and .BR check assertions that you make; they do no checking at run time.

.BR addr

where:

addr is the address you want to test.

.CALLK

EMT 373

The .CALLK request transfers control (and alters mapping) from the current mode to the specified virtual address in Kernel mode.

.CALLK [dest][,pic]

where:

dest	is a virtual address in Kernel mode; the address of the entry point to the routine. If *dest* is not specified, .CALLK assumes the address is on the stack
pic	is an optional parameter that should be non-blank for PIC.

The environment upon entry into Kernel mapping is as follows:

- Registers 0-5 are preserved.
- The PS is not preserved.
- The contents of the User and Kernel stacks are undefined.
- The User mapped system communications area (SYSCOM) is not mapped to Kernel.

The routine called in Kernel mode must return, using the standard RETURN instruction. The environment upon return to User mapping is:

- Registers 0-5 are preserved across the change to User mode.
- The condition codes in the PS are preserved.
- The trace trap bit and previous mode bit in the PS are not preserved.
- The stack pointer (SP) and stack contents are the same as before the call to .CALLK, except that the destination address has been popped off the stack.

Errors:
None reported by .CALLK; however, the called routine may report errors.

.CALLS

Fully-Mapped Only

The .CALLS request supports transferring control to Supervisor mode. It is designed to work with the SHANDL Supervisor handler code.

.CALLS dest,return

where:

dest Supervisor virtual address (#xxxxxx) to which you are transferring control.

return Character string designating condition code values that should be preserved for return transition from Supervisor mode to User mode. Default value is NZVC, meaning return all condition codes.
If no condition codes need to be returned, specify RETURN=<>. If only carry needs to be returned, specify RETURN=C. Any combination of condition codes may be specified.

.CALLS is used as a transfer vector, not as an inline call.

.CALLS also has a special form, .CALLS #0, by which you can transfer control to Supervisor mode and perform an RTI instruction (with Supervisor mode as the "previous" mode).

.CDFN

EMT 375, Code 15

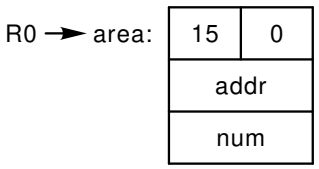
The .CDFN request redefines the number of I/O channels. .CDFN allows the number to be expanded to as many as 255(decimal) channels.

The space for the new channels must be allocated by the User program. Allocate 5*n words of memory, where *n* is the number of channels to be defined.

.CDFN area,addr,num

where:

area	is the address of a three-word EMT argument block
addr	is the address where the I/O channels begin
num	is the number of I/O channels to be created



Errors:

Code	Explanation
0	An attempt was made to define fewer than or the same number of channels that already exist. Violation of PAR1 restriction.

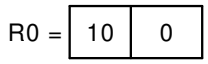
.CHAIN

EMT 374, Code 10

The .CHAIN request lets a background program pass control directly to another background program without operator intervention. Since this process can be repeated, a long "chain" of programs can be strung together.

The chain area consists of locations 500-777 of the running job's virtual address space.

.CHAIN



Errors:
.CHAIN is implemented by simulating the monitor RUN command and can produce any errors that RUN can produce. If an error occurs, .CHAIN is abandoned and the keyboard monitor is entered.

.CHCOPY

Multijob Only

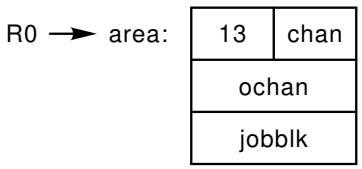
EMT 375, Code 13

The .CHCOPY request opens a channel for input, logically connecting it to a file that is currently open by another job. This request must be issued before the first .READ or .WRITE request on that channel.

.CHCOPY area,chan,ochan [,jobblk]

where:

area	is the address of a three-word EMT argument block
chan	is the channel the current job will use to read the data
ochan	is the channel number of the other job's channel to be copied
jobblk	is a pointer to a three-word ASCII logical job name that represents a system job



Errors:

Code	Explanation
0	Other job does not exist, does not have enough channels defined, or does not have the specified channel *ochan* open.
1	Channel *chan* already open.

.CKXX

The .CKXX macro generates CK.Rn register checking macros.

.CKXX <reg[,alph][,reg,...]>

where:

reg	is the register or registers you want .CKXX to define as check registers.

The check macro CK.Rn, generated by .CKXX, has the following form:

CK.Rn[alph] [label][,change][,result]

where:

n	is the register number that .CKXX assigned to this check macro
alph	is an alphabetic character that .CKXX assigned to this check macro
label	is the value or label you assume equates to the check register.
change	indicates the check macro increment or decrement. The change value must be preceded by a plus sign (+) to indicate increment or a minus sign (-) to indicate decrement.
result	is a new value assigned by the check macro.

.CLOSE

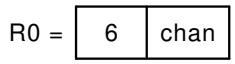
EMT 374, Code 6

The .CLOSE request terminates activity on the specified channel and frees it for use in another operation.

.CLOSE chan

where:

chan is a channel number in the range 0 to 376_8



Errors:

Code	Explanation
3	A protected file with the same name already exists on the device. The .CLOSE is performed anyway, resulting in two files with the same name on the device.

.CLOSZ

EMT 375, Code 45

The .CLOSZ programmed request terminates activity on a channel that was opened by a .ENTER, frees it for use in another operation, and sets the file size.

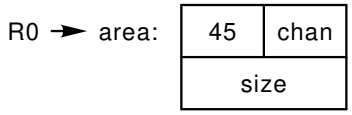
.CLOSZ area,chan,size

where:

area is the address of a 2-word EMT argument block.

chan is a channel number in the range of 0 to 376_8.

size is the specified size of the file at closing.



Errors: (If channel was opened to an RT–11 *directory device*)

Code	Explanation
1	Size argument is greater than allocated size; file closed at size indicated by highest block written (equivalent to .CLOSE)
2	Channel not opened with .ENTER; channel purged
3	A protected file with the same name already exists on the device. File is closed with size as indicted by size argument. (If error 1 and error 3 conditions exist at same time, error 1 takes precedence)

Errors: (If channel was opened on a *special directory device*)

Code	Explanation
1	Meaning controlled by handler
2	Channel not opened by .ENTER; channel purged
3	Meaning controlled by handler

.CMAP

EMT 375, Code 46, Subcode 0

The .CMAP request sets the "CMAP" status and returns the old value in R0.

.CMAP area,value,CODE=strg

where:

value	is the setting desired
CODE=strg	Specify *strg* as either "SET" (default), "NOSET", "SP" or "STACK"

.CMKT

Timer Support

EMT 375, Code 23

The .CMKT request causes one or more outstanding mark time requests to be canceled (See the .MRKT programmed request).

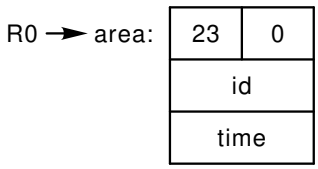
.CMKT area,id[,time]

where:

area is the address of a three-word EMT argument block

id is a number that identifies the mark time request to be canceled. If more than one mark time request has the same *id*, the request with the earliest expiration time is canceled. If *id* = 0, all non-system mark time requests (those in the range 1 to 176777) for the issuing job are canceled

time is the address of a two-word area in which the monitor returns the amount of time (clock ticks) remaining in the canceled request. If an address of 0 is specified, no value is returned. If id = 0, the time parameter is ignored and need not be indicated



Errors:

Code	Explanation
0	The *id* was not zero and a mark time request with the specified identification number could not be found (implying that the request was never issued or that it has already expired).

.CNTXSW

EMT 375, Code 33

The .CNTXSW request is used to specify a list of locations saved when a context switch occurs.

If .CNTXSW is issued more than once, only the latest list is used; the previous address list is discarded. If the address *addr* is 0, no extra locations are switched. The .CNTXSW request is ignored for virtual jobs,.

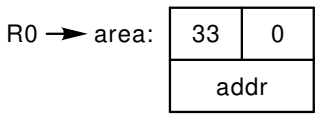
.CNTXSW area,addr

where:

area is the address of a two-word EMT argument block

addr is a pointer to a list of addresses terminated by a zero word. The addresses must be even and be:

- in the range 2-476
- in the user job area
- in the I/O page



Errors:

Code	Explanation
0	One or more of the conditions specified by *addr* was violated.

.CRAW

Mapped Only

EMT 375, Code 36, Subcode 2

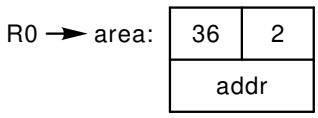
The .CRAW request defines a virtual address window. Optionally, mapping occurs if you set the WS.MAP bit in the last word of the window definition block before you issue .CRAW.

.CRAW area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of the window definition block. (See .WDBBK)



Errors:

Code	Explanation
0	Window alignment error: the new window overlaps the static window for a virtual job. The window is too large or W.NAPR is greater than 7.
1	An attempt was made to define more than seven windows in your program. Eliminate a window (.ELAW) or redefine your virtual address space into fewer windows.

If the WS.MAP bit were set in the window definition block status word:

Code	Explanation
2	An invalid region identifier was specified.
4	The combination of the offset into the region and the size of the window to be mapped into the region is invalid.
17	Inactive mode or space specified.

.CRRG

Mapped Only

EMT 375, Code 36, Subcode 0

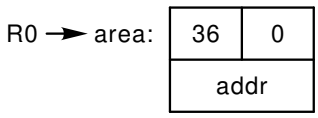
The .CRRG request directs the monitor to allocate a dynamic region in physical memory for use by the current requesting program.

.CRRG area[,addr]

where:

area is the address of a two-word EMT argument block

addr is the address of the region definition block. (See .RDBBR)



Errors:

Code	Explanation
6	No region control blocks are available.
7	A region of the requested size cannot be created because not enough memory is available. The size of the largest available region is returned in R0.
10	An invalid region size was specified. A value of 0, or a value greater than the available amount of contiguous extended memory, is invalid.
12	Global region not found.
13	Too many global regions in use (none free).
15	Global region is privately owned.
16	Global region already exists at a different base address.

.CSIGEN

EMT 344

The .CSIGEN request processes a standard RT–11 command string. In general mode, file .LOOKUP and .ENTER requests as well as handler .FETCH requests are performed. This request returns information on the stack.

When called in general mode, the CSI purges channels 0 through 10_8 before processing the command string. If errors occur during processing, it purges them again.

.CSIGEN devspc,defext[,cstrng][,linbuf]

where:

devspc is the address of the memory area where the device handlers (if any) are to be loaded.

defext is the address of a four-word block that contains the Radix–50 default file types.

cstrng is the address of the ASCIZ command string or a 0 if input is to come from the console terminal.

linbuf is the storage address of the original command string. This is a user-supplied area, 81_{10} bytes in length.

On return, R0 points to the first available location above the handlers, the stack contains the option information, and all the specified files have been opened.

Errors:

Code	Explanation
0	Invalid command (such as bad separators, invalid file names, and commands that are too long).
1	A device specified is not found in the system tables.
2	A protected file of the same name already exists. A new file was not opened.
3	Device full.
4	An input file was not found in a .LOOKUP.

Passing Option Information

Both .CSIGEN and .CSISPC return options and their associated values in reverse order from that specified on the command line. That is, the last option and associated value (if present) placed last on the stack will be the first option retrieved.

Options and their associated values are returned on the stack. The format of the stack output of the CSI for options is as follows:

Word	Value	Meaning
1	N	Number of options found in command string. If N=0, no options were found.
2	Option character and file number	Even byte = seven-bit ASCII option character Bits 8-14 = file number Bit 15 = 1 if the option had a value
3	Option value	If bit 15 of word 2 is set, word 3 contains the option value. If bit 15 is not set, word 3 is omitted. Words 2 and 3 repeat n times.

.CSISPC

EMT 345

The .CSISPC request calls the Command String Interpreter to parse the command string and return file descriptors and options to the program. The CSI does not perform any .CLOSE, .ENTER, .LOOKUP, or handler .FETCH requests.

Options and their associated values are returned on the stack. The optional argument *linbuf* can provide your program with the original command string.

.CSISPC outspc,defext[,cstrng][,linbuf]

where:

outspc is the address of the 39-word block to contain the file descriptors.

defext is the address of a four-word block that contains the Radix–50 default file types.

cstrng is the address of the ASCIZ input string or a #0 if input is to come from the console terminal.

linbuf is the storage address of the original command string. This is a user-specified area, 81 bytes in length.

Notes

- The file description consists of 39 words, comprising nine file descriptor blocks (five words for each of three possible output files; four words for each of six possible input files), which correspond to the nine possible files (three output, six input). If any of the nine possible file names are not specified, the corresponding descriptor block is filled with zeroes.
- The five-word blocks hold four words of Radix–50 representing *dev:file.type*, and one word representing the size specification given in the string.

Errors:

Code	Explanation
0	Invalid command line.
1	Invalid device.

.CSTAT

EMT 375, Code 27

This request furnishes you with information about a channel.

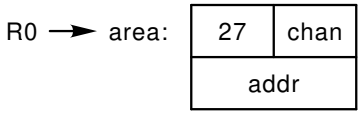
.CSTAT area,chan,addr

where:

area is the address of a two-word EMT argument block

chan is the number of the channel about which information is desired

addr is the address of a six-word block to contain the status.



Errors:

Code	Explanation
0	The channel is not open.

.CTIMIO

Timeout

Macro Expansion

The .CTIMIO macro cancels the device time-out request in the handler interrupt service section when an interrupt occurs to disable the completion routine (See .TIMIO).

.CTIMIO tbk

where:

tbk is the address of the seven-word timer block.

Errors:
None.

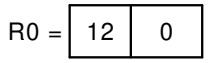
.DATE

EMT 374, Code 12

This request returns in R0 the current date information from the system date word.

.DATE

where:



Errors:
No errors are returned. A zero result in R0 indicates that the user has not entered a date.

.DEBUG

.DEBUG sets up the environment for the .DPRINT macro and may generate routines to support octal/decimal displays used by .DPRINT. Before .DEBUG is issued, .ENABL LSB needs to be in effect.

.DEBUG switch,class,pic,id,value,psect,code,?L1,?L2,?L3,?L4

where:

switch	ON/on	Set V23 to value of CLASS
	OFF/off	Set V23 to 0 (default)
class	177777	Select all classes
	xxxxxx	Bit mask to select classes
pic	Default	Do not generate PIC code
	YES	Generate PIC
id	Default	Do not generate separated I-D code
	YES	Generate separated I-D code
value	Default	No value printing
	YES	Value printing
psect	(Deb$ug)	psect used for text string generated by .DPRINT
code	(Deb.ug)	psect used for support subroutines
L1–L4	(xxxxx$)	Local labels used for support routines

.DELETE

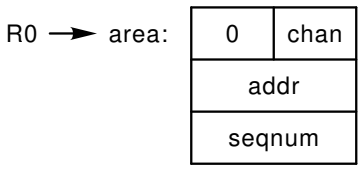
EMT 375, Code 0

The .DELETE request deletes a specified file from a specified device. The .DELETE request is invalid for magtapes.

.DELETE area,chan,dblk[,seqnum]

where:

area	is the address of a three-word EMT argument block
chan	is the device channel number
dblk	is the address of a four-word Radix–50 descriptor
seqnum	is a file position number (Not supported or used by any Digital-supplied handler)



Errors:

Code	Explanation
0	Channel is not available.
1	File was not found in the device directory.
2	Invalid operation.
3	The file is protected and cannot be deleted.

.DEVICE

EMT 375, Code 14, Subcodes 0, 1

You set up the list of addresses with the specified values. When your program exits the system loads these designated addresses with the corresponding values.

The .DEVICE request is ignored when it is issued by a virtual job.

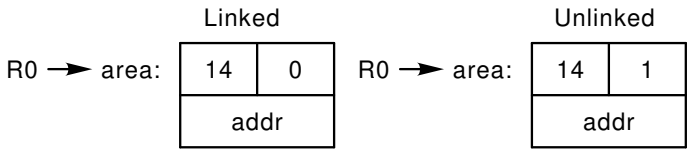
.DEVICE area,addr[,link]

where:

area is the address of a two-word EMT argument block

addr is the address of a list of two-word elements. Each element of a list is composed of a one-word address and a one-word value to be put at that address. If *addr* is #0, any previous list is discarded.

link is an optional argument that, if present, specifies linking of tables on successive calls to .DEVICE. If the argument is omitted, the list referenced in the previous .DEVICE request is replaced by the new list. The argument must be supplied to cause linking of lists; however, linked and unlinked list types cannot be mixed



Errors:
None.

.DPRINT

This macro conditionally generates code to print a string.

.DPRINT print[,value][,type][,class],?L1

where:

value Value to print if non-blank.
Use R0 to print value in R0
Avoid stack references.
Note: .DPRINT issues a .PRINT that clears location 52, the error byte, before .DPRINT picks up any value to display. To display contents of the error byte, move it to a temporary location, and reference that location.

type (OCT) Display in octal
(DEC) Display in decimal

class (177777) Generate code if any class enabled
xxxxxx Generate code if ...V23 and class is non-zero

L1 (xxxxx$) Local symbol. If .ENABLE LSB is not in effect, you must provide a symbol for L1.

.DRAST

Macro Expansion (Handlers Only)

The .DRAST macro sets up the interrupt and abort entry points, lowers the processor priority, and references global symbol $INPTR, which contains a pointer to the $INTEN routine in the resident monitor.

.DRAST name,pri[,abo]

where:

name	is the two-character device name
pri	is the priority of the device, and also the priority at which the interrupt service code is to execute
abo	is an optional argument that represents the label of an abort entry point. If you omit this argument, the macro generates a RETURN instruction at the abort entry point, which is the word immediately preceding the interrupt entry point.

.DRBEG

Macro Expansion (Handlers Only)

.DRBEG sets up a variable number of words (at least six) as the first words of the handler. The number of words set up by .DRBEG is determined by options selected in this and other .DRxxx macros. This macro also generates the appropriate global symbols for your handler. Before you use .DRBEG, you must invoke .DRDEF to define xxCSR, xxVEC, xxDSIZ, and xxSTS (See macro .DRDEF).

.DRBEG name[,SPFUN=spsym][,NSPFUN=nspsym]

where:

name is the 2-character device name.

spsym is the symbol name for the list of DMA standard special functions.

nspsym is the symbol name for the list of DMA nonstandard special functions.

The arguments, *spsym* and *nspsym*, point to lists of special functions within the low-memory portion of the handler. The special functions are listed in the same manner as that used by the .DRSPF macro *extension table* method for defining special functions. Standard DMA special functions are listed in a group (assigned a symbol name) and that symbol name is used by .DRBEG (*spsym*). Nonstandard DMA special functions are listed in a separate group with a different symbol name and that name is then used by .DRBEG (*nspsym*).

.DRBOT

Macro Expansion (Handlers Only)

.DRBOT generates a routine that issues a read request to read blocks 3-5 from the device into memory, an error message routine used to report errors during the booting process, and a structure in which you write the bootstrap read routine.

.DRBOT name,entry,read
[,CONTROL=arg...,arg][,SIDES=n][,force][,PSECT]

where:

name	is the two-character device name
entry	is the software bootstrap entry point
read	is the bootstrap read entry point
CONTROL	defines the types of controllers supported by this handler. The values can be UBUS or QBUS. If CONTROL is omitted, both Unibus and Q-bus are assumed. This is correct for all supported handlers
SIDES	specifies single- or double-sided diskettes. If omitted, single-sided are assumed.
force **PSECT**	Both of these arguments are passed to a .DREND macro automatically generated by .DRBOT.

.DRDEF

Macro Expansion (Handlers Only)

The .DRDEF macro sets up handler parameters, calls the driver macros from the library, and defines useful symbols.

.DRDEF name,code,stat,size,csr,vec [,UNIT64=str][,DMA=str][,PERMUMR=n] [,SERIAL=str]

where:

name is the two-character device name.

code is the device identifier numeric code

stat is the device status bit pattern. (See Data Structures)

size is the size of the device in 256-word blocks.
If the device is not random access, use 0.

csr is the default address for the device's CSR.

vec is the default vector.

UNIT64 indicates whether this handler supports extended device units:

- **No** Default. No extended units
- **Yes** Extended units

DMA indicates whether this handler supports direct memory access:

- **Yes** Use direct memory access
- **No** No direct memory access.

PERMUMR indicates this handler should be assigned *n* permanent UNIBUS mapping registers. Valid values for *n* are 0 through 7. The default is 0.

SERIAL indicates whether this handler requires serialized I/O request satisfaction:

- **No** Default. No serialized I/O request satisfaction
- **Yes** Requires serialized I/O request satisfaction

.DRDEF

Device-Identifier Byte

The low byte of the device status word, the device-identifier byte, identifies each device in the system. You specify the correct device identifier as the *code* argument to .DRDEF. The values are defined in octal as Table 5–4 shows.

Table 5–4: Device-Identifier Byte Values

Name	Code	Device
RK	0	RK05 Disk
	1	Reserved
EL	2	Error Logger
LP	3	Parallel Interface Printer
BA	4	Batch Handler
DL	5	RL01/RL02 Disk
DY	6	RX02 Diskette
	7	Reserved
VS	10	RTEM Virtual System VS(M)
MT	11	TM11/TMA11/TU10/TS03 Magtape
	12-14	Reserved
IB	15	IVB11 Bus Driver
	16–17	Reserved
MM	20	TJU16/TU45 Magtape
	21	Reserved
DX	22	RX11/RX01 Diskette
DM	23	RK06/RK07 Disk
	24	Reserved
NL	25	Null Device
	26–30	Reserved (DECnet)

Table 5–4 (Cont.): Device-Identifier Byte Values

Name	Code	Device
	31–33	Reserved (CTS–300)
	34	Reserved
MS	35	TS11/TS04/TS05 Magtape
	36–40	Reserved
LS	41	Serial Interface Printer
MQ	42	Internal Message Handler
DR	43	DRV11J Interface (MRRT)
XT	44	Reserved (MRRT)
	45	Reserved
LD	46	Logical Disk Handler
VM	47	KT11 Pseudodisk Handler
DU	50	MSCP Disk Class Handler
SL	51	Single-line Command Editor
DZ	52	RX50 Diskette (CTI Bus-based processor)
DW	53	Hard Disk (CTI Bus-based processor)
PI	54	Professional interface
SP	55	Transparent spooler
	56	Reserved
XC/XL	57	Communication port (Professional 325/350 or DL(V)–11)
MU	60	TMSCP Magtape Class handler
NC/NQ/NU	61	Ethernet Class handler

Table 5–4 (Cont.): Device-Identifier Byte Values

Name	Code	Device
SD	62	DBG–11 Handler
ST	63	DBG–11 Symbol Table Handler
	64	Reserved
UB	65	UMR Pseudohandler

.DREND

Macro Expansion (Handlers Only)

The .DREND macro generates the table of addresses for service routines in RMON.

.DREND name[,FORCE=n][,PSECT=psect]

where:

name is the two-character device name

FORCE Value supplied as FORCE, combined with the settings of MMGT, ERLG, and TIM$IT, selects the entries to be generated in the vector table.

Values in FORCE correspond to SYSGEN options. The default value for FORCE is 0. Using a value of -1 for FORCE will generate all the possible entries of the System Service Table.

PSECT forces the .DREND request to be placed in the specified program section at link time. Use this argument when the handler is built from several PSECTs.

.DREST

Macro Expansion (Handlers Only)

The .DREST macro places device specific information in block 0 of a device handler. The device specific information includes:

- The device class
- The variants of a device class and additional information about some device classes
- Whether the device handler contains updatable internal data table(s) accessible by SPFUN 372
- The type (device class) of the updatable internal data table
- Whether the device handler has a table in block 0 that contains bad-block replacement information
- How the handler can be installed, loaded, and mounted

.DREST [CLASS=n][,MOD=n][,DATA=dptr]
[,TYPE=n][,SIZE=n][,REPLACE=rptr][,MOD2=n][,STAT2=symb]

where:

CLASS is the device class:

Symbol	Value	Meaning
DVC.CT	6	Cassette tape
DVC.DE	10	DECnet executive pseudohandler
DVC.DK	4	RT–11 file structured disk (DD, DL, DM, DP, DT, DU, DW, DX, DY, DZ, LD, RK, VM)
DVC.DL	12	DECnet port (line) handler
DVC.DP	11	DECnet protocol pseudohandler
DVC.LP	7	Printer (LP, LS, SP)
DVC.MT	5	Magtape (MM, MS, MT, MU)
DVC.NI	13	Ethernet port handler (NC, NQ, NU)
DVC.NL	1	NULL handler (AT and NL)
DVC.PS	14	Pseudohandler (PI, SL, UB)

DVC.SB	20	Serialized input/output (PC and generic)
DVC.SI	16	Serialized input (generic)
DVC.SO	17	Serialized output (generic)
DVC.TP	3	Reserved
DVC.TT	2	Terminal class handler (BA and TT)
DVC.UK	0	Unknown device class
DVC.VT	15	Virtual terminal port handler (XL, XC) Values in the range of 200 through 377 are reserved for the user.

MOD indicates a variation or additional information about a device class.

(None)	0	No variant or information (default)
DVM.DM	2	With CLASS=DVC.DK, indicates device has an extra error word prefixed to SPFUN 376 and SPFUN 377 buffers
DVM.DX	1	With CLASS=DVC.DK, indicates device is an RX01-compatible drive
DVM.NS	1	With CLASS=DVC.MT, indicates file handler has no file structure support
DVM.NF	200	With all class devices, indicates handler can only be loaded and cannot be fetched. This bit is read-only and cannot be set using the .DREST macro. (This bit is set by the .DRPTR macro with the FETCH=*NO* argument.)
DVM.NL	100	With all class devices, indicates handler cannot be loaded. Bit DVM.NL is set by the .DRPTR macro LOAD=*NO* argument.

DATA specifies whether the handler has internal updatable data table(s) accessible by SPFUN 372. For the *DATA* argument, *dptr* can be:

0	Default. No internal data table
dptr	is the file address of the internal data table(s).

.DREST

TYPE specifies whether a device type classification exists for the internal data table(s). For the *TYPE* argument, *n* can be:

omitted	Default; no device type classified internal table
rad	RAD50 device type classification

SIZE Provides size of table (pointed to by *DATA*)

REPLACE specifies whether the handler has a table in block 0 that contains bad-block replacement geometry information. For the *REPLACE* argument, *rptr* can be:

0	Default; no bad-block replacement geometry table
rptr	Is the file address of a bad-block replacement geometry table.

MOD2 Currently only supports the LS handler:

DV2.V2	First .DRVTB table is followed by second display only.

STAT2 specifies the conditions under which the handler can be installed, loaded, and mounted. The *STAT2* argument, *symb* can be:

0	Default. .DREST specifies no restrictions.
HS2.BI	Cannot be installed by the monitor.
HS2.KI	Cannot be installed by the INSTALL command
HS2.KL	Cannot be loaded by the LOAD command.
HS2.KU	Cannot be unloaded by the UNLOAD command
HS2.MO	Can be mounted by the MOUNT command and dismounted by the DISMOUNT command

Errors:
None

.DRFIN

Macro Expansion (Handlers Only)

The .DRFIN macro generates the instructions for the jump back to the monitor at the end of the handler I/O completion section. The macro makes the pointer to the current queue element a global symbol, and it generates position-independent code for the jump to the monitor. When control passes to the monitor after the jump, the monitor releases the current queue element.

.DRFIN name

where:

name is the two-character device name

Errors:
None.

.DRINS

Macro Expansion (Handlers Only)

The .DRINS macro defines the following:

- Symbols that contain the CSR addresses listed by RESORC (display CSRs) and the CSR checked by the INSTALL keyboard command.
- Separate entry points for installing the handler as a system device or as a data device.
- List of CSR addresses in block 0.

.DRINS name,<csr,csr,...>

where:

name The two-letter device mnemonic.

– name Specifying CSR = *NO* to the .DRDEF macro prevents it from filling in INSCSR:

- Fill the value DISCSR
- Do not fill the value INSCSR

csr Specifies a symbolic CSR address for that device. If more than one display CSR exists, separate them with commas and enclose the list in angle brackets <>. With multiple display CSRs (first CSR is offset 176, second CSR is offset 174...), you do not have to list the first CSR.

.DRPTR

Macro Expansion (Handlers Only)

The .DRPTR macro places pointers to handler service routines.

.DRPTR [FETCH=n][,RELEASE=n][,LOAD=n][,UNLOAD=n]

where:

FETCH specifies whether a handler service routine is called by the .FETCH programmed request. For the *FETCH* argument, *n* can be:

0 Default. No .FETCH service routine.

n Address of the .FETCH service routine.

NO A literal string; the handler cannot be fetched.

RELEASE specifies whether a handler service routine is called by the .RELEASE programmed request. For the *RELEASE* argument, *n* can be:

0 Default. No .RELEASE service routine.

n is the file address of the service routine to be called by .RELEASE.

LOAD specifies whether a handler service routine is called when the handler is loaded by bootstrap routine or LOAD command. For the *LOAD* argument, *n* can be:

0 Default. No *load* service routine.

n Address of the *load* service routine.

NO Literal string. Handler cannot be loaded.

UNLOAD specifies whether a handler service routine is called when the handler is unloaded by the UNLOAD command. For the *UNLOAD* argument, *n* can be:

0 Default. No *unload* service routine.

n Address of *unload* service routine.

Errors:
None.

.DRSET

Macro Expansion (Handlers Only)

The .DRSET macro sets up the option table for the SET command in block 0 of the device handler file. Use this macro once for each SET option that is used.

.DRSET option,val,rtn[,mode]

where:

option	is the name of the SET option, such as WIDTH or CR. Up to six alphanumeric characters. No embedded spaces or tabs
val	is a parameter that is passed to the routine in R3. Must not be zero.
rtn	is the name of the routine that modifies the code in block 1 of the handler.
mode	is an optional argument to indicate the type of SET parameter

A *NO* indicates that a NO prefix is valid for the option. NUM indicates that a decimal numeric value is required. OCT indicates that an octal numeric value is required. Omitting this argument indicates that the option takes neither a NO prefix nor a numeric argument. NO may be combined with NUM or OCT.

Errors:
None.

.DRSPF

Macro Expansion (Handlers Only)

The .DRSPF macro defines the special function codes supported by a handler.

.DRSPF arg[,arg2][,TYPE=n]

where:

arg can be specified in two ways: the *list* method and the *extension table* method

.

arg2 is a list of special function codes. Only use the *arg2* argument to specify special function codes in an extension table, that is, when the *arg* argument is a minus sign (-). See the discussion of the *arg* argument in the extension table method description.

TYPE=n Type of special function or functions that are coupled with this parameter. See Table 5–5 for valid symbols, bit pattern for bits 8, 9, and 10, and the meaning for the *n* argument.

Table 5–5: Special Functions for the TYPE=n Parameter

Symbol	Bit	Meaning
O	000	The letter O (default) indicates unknown function code.
R	001	A read operation special function code. Any operation that obtains data from a device is defined here as a read operation.
W	010	A write operation special function code. Any operation that directs data to a device is defined here as a write operation.
M	011	A motion operation special function code. Any operation whose sole purpose is to cause the device to move is defined here as a motion operation.
T	100	A read/write operation special function code. The sign of the bit for the special function word count (wcnt) parameter determines if the operation is a read or a write.
	101-111	Reserved for Digital.

.DRSPF

List Method

Arg can be a list of one or more special function codes. That list is located in block 0 of the handler at locations 22 through 27.

Extension Table Method

Arg can be a pointer to an extension table address. Do not place the extension table in block 0 of the handler.

The pointer to the extension table address must be prefixed by a plus sign (+). The extension table address must have the high bit cleared.

The extension table contains one or more .DRSPF macros. The *arg* argument for each .DRSPF macro is a minus sign (-).

Errors:
None.

.DRTAB

Macro Expansion (Handlers Only)

The .DRTAB macro establishes the file address of a list of handler data tables for Digital use only. A similar macro, .DRUSE, is available for user-defined handler data tables.

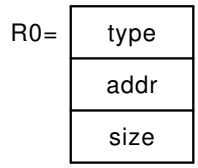
The relationship between .DRTAB and .DREST macros is:

- When a distributed handler contains only one handler data table, .DREST is used to describe that table.
- When a distributed handler contains more than one handler data table, the .DRTAB macro is used to describe all those tables.

.DRTAB type,addr,size

where:

type	is the handler data table format name in one to three RAD50 characters
addr	is the file address
size	is the size in bytes



.DRTAB is invoked once for each handler data table. Each invocation of .DRTAB creates a 3-word descriptor containing the values specified for the *type, addr,* and *size* arguments. A call to .DRTAB with no arguments specifies the end of that list of handler data table descriptors.

.DRTAB places the file address of the list of handler data table descriptors in block 0 of the handler. The list of descriptors and the data tables themselves are not located in block 0. When first invoked, .DRTAB sets up locations 70 through 74 in block 0.

.DRUSE

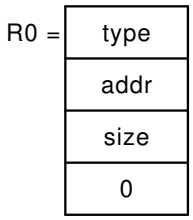
Macro Expansion (Handlers Only)

The .DRUSE macro establishes the file address of a list of user-defined handler data tables.

.DRUSE type,addr,size

where:

type	is a handler data table format name in one to three RAD50 characters
addr	is the file address of the handler data table
size	is the size in bytes of the handler data table



Invoke .DRUSE once for each user-defined handler data table in your handler. Each invocation of .DRUSE creates a 3-word descriptor containing the values you specified for the *type, addr*, and *size* arguments. Call .DRUSE with *no* arguments to indicate the end of the list of descriptors.

.DRUSE places the file address of the list of handler data table descriptors in block 0 of your handler. Do not place the list of descriptors or the data tables themselves in block 0 of your handler. When you first invoke .DRUSE, it sets up location 106 in block 0 with the following contents:

Location	Contents After .DRUSE Is Invoked
106	Pointer to list of handler data table descriptors

Errors:
None.

.DRVTB

Macro Expansion (Handlers Only)

The .DRVTB macro sets up a table of three-word entries for each vector of a multivector device. The table entries contain the vector location, interrupt entry point, and processor status word. You must use this macro once for each device vector.

.DRVTB name,vec,int[,ps]

where:

name is the two-character device name. This argument must be blank except for the first-time use of .DRVTB

vec is the location of the vector, and must be between 0 and 474

int is the symbolic name of the interrupt handling routine. It must appear elsewhere in the handler code. It generally takes the form ddINT, where dd represents the two-character device name

ps is an optional value that specifies the low-order four bits of the new Processor Status Word in the interrupt vector. This argument defaults to zero if omitted. The priority bits of the PS are set to 7, even if you omit this argument

Errors:
None.

.DSTAT

EMT 342

This .DSTAT request obtains information about a particular device.

.DSTAT retspc,dnam

where:

retspc is the address of a four-word block that stores the status information

dnam is the address of a word containing the Radix–50 device name.

.DSTAT looks for the device specified by *dnam* and, if successful, returns four words of status starting at the address specified by *retspc*:

Word 1 Status Word

Bits 0-7: The low-order byte contains a numeric code (.DEVDF) that is the device identifier value for the device in the system.

Bits 8-15: The low-order byte contains value for stat (.DSTDF), a device status bit pattern returned from the .DRDEF request. Value symbols:

FILST$ = 100000 HNDLR$ = 4000

RONLY$ = 40000 SPFUN$ = 2000

WONLY$ = 20000 ABTIO$ = 1000

SPECL$ = 10000 VARSZ$ = 400

Word 2 Handler Size

The size of the device handler in bytes.

Word 3 Load Address +6

Non-zero handler is in memory. Address is handler load address +6.
Zero implies that it must be fetched before it can be used.

Word 4 Device Size

The size of the device (in 256-word blocks) for block-replaceable devices; 0 for sequential-access devices, the smallest-sized volume for variable-sized devices. The last block on the device is the device size -1.

Errors:

Code	Explanation
0	Device not found in tables.

.ELAW

Mapping Only

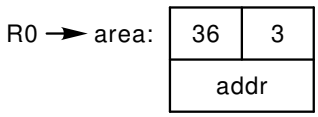
EMT 375, Code 36, Subcode 3

The .ELAW request eliminates a virtual address window. An implicit unmapping of the window occurs when its definition block is eliminated.

.ELAW area,addr

where:

area	is the address of a two-word EMT argument block
addr	is the address of the window definition block for the window to be eliminated



Errors:

Code	Explanation
3	An invalid window identifier was specified.
17	Inactive mode or space was specified.

.ELRG

Mapping Only

EMT 375, Code 36, Subcode 1

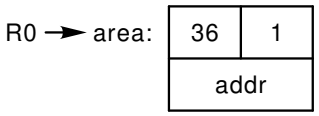
The .ELRG request directs the monitor to eliminate a dynamic region in physical memory and return it to the free list where it can be used by other jobs.

.ELRG area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of the region definition block for the region to be eliminated. Windows mapped to this region are unmapped. The static region cannot be eliminated



Errors:

Code	Explanation
2	An invalid region identifier was specified.
14	Global region in use.
11	Deallocation failure.

Example:
Refer to example in the *RT–11 System Internals Manual*.

.ENTER

EMT 375, Code 2

The .ENTER request allocates space on the specified device and creates a tentative entry with the name of the specified file. The channel number specified is associated with the file.

.ENTER area,chan,dblk,len[,seqnum]

where:

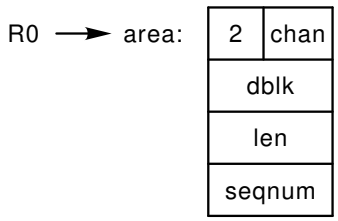
area is the address of a four-word EMT argument block

chan is a channel number.

dblk is the address of a Radix–50 descriptor of the file.
If the file name is not provided in *dblk*, it is a non-file-structured .LOOKUP which connects the channel to the entire device, starting at block 0.

len is the file size specification.

seqnum is a parameter for magtape. See *RT–11 Device Handlers Manual*.



On return from this call, R0 contains the size of the area actually allocated for use. Or zero (0) for a non-RT–11 device or non-file-structured .ENTER.

Errors:

Code	Explanation
0	Channel is not available.
1	In a fixed-length request, no space greater than or equal to *m* was found; or the device or the directory was found to be full.
2	Nonshareable device is already in use.
3	A file by that name already exists and is protected. A new file was not opened.
4	File sequence number was not found.
5	File sequence number is invalid or file name is null.
6	Nonexistent or otherwise invalid special-directory device unit. The handler determines the validity of the device unit.

.EXIT

EMT 350

.EXIT

The .EXIT request causes the user program to terminate. When used from a background job, .EXIT causes KMON to run in the background area, prior to running KMON; all outstanding mark-time requests are canceled; and I/O requests and/or completion routines pending for that job are allowed to complete.

If R0 = 0 when the .EXIT is done, an implicit .HRESET is executed when KMON is entered, disabling the subsequent use of REENTER, START or CLOSE. See .HRESET.

The .EXIT request enables a user program to pass command lines to KMON in the chain information area (locations 500-777_8) for execution after the job exits. This is performed under the following conditions:

- The word (not byte) location 510 must contain the total number of bytes of command lines to be passed to KMON.
- The command lines are stored, beginning at location 512. The lines must be .ASCIZ strings with no embedded carriage return or line feed.

  The user program must set SPXIT$ or CHNIF$ in the Job Status Word before doing an .EXIT, which must be issued with R0 = 0.

.FETCH

EMT 343

The .FETCH request loads device handlers into memory from the system device.

.FETCH addr,dnam

where:

addr	is the starting address at which the device handler is to be LOADed
dnam	is the pointer to the Radix–50 device name

Errors:

Code	Explanation
0	The device name specified is not installed and there is no logical name that matches the name and the *catch-all has not been assigned.

.FORK

Macro Expansion

The .FORK call is used when access to a shared resource must be serialized or when a lengthy but non-time-critical section of code must be executed. .FORK issues a subroutine call to the monitor and does not use an EMT instruction request.

.FORK fkblk

where:

fkblk is a four-word block of memory allocated within the driver

Errors:
None.

The .FORK macro expands as follows:

```
.TITLE  EFORK1.MAC

        .FORK   fkblk

        JSR     R5,@$FKPTR
        .WORD   fkblk-.
```

The .FORK call must be preceded by an .INTEN call. Your program must not have left any information on the stack between the .INTEN and the .FORK calls. The contents of registers R4 and R5 are preserved through the call and, on return, registers R0 through R3 are available.

If you are using a .FORK call from a device handler, it is assumed that you used .DREND provided for handlers. The .DREND macro allocates a word labeled $FKPTR.

If you want to use the .FORK macro in an in-line interrupt service routine rather than in a device handler, you must set up $FKPTR.

.FPROT

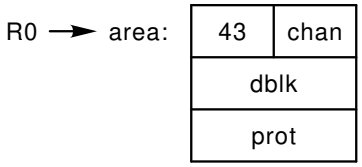
EMT 375, Code 43

The .FPROT programmed request sets or removes file protection on individual RT–11 files. A file marked as protected cannot be deleted by .CLOSE, .DELETE, .ENTER, or .RENAME.

.FPROT area,chan,dblk[,prot]

where:

area	is the address of a three-word EMT argument block
chan	is a channel number.
dblk	is the address of a four-word block containing the filespec in Radix–50.
prot	#1 (or omitted) to protect. #0 to unprotect.



Errors:

Code	Explanation
0	Channel is not available.
1	File not found or not a file-structured device.
2	Invalid operation.
3	Invalid value for PROT.

.FPROT returns the previous file status word in R0:

- If the high bit in R0 is off, the file was not previously protected.
- If the high bit in R0 is on, the file was previously protected.

.GCMAP

Fully Mapped Only

EMT 375, Code 46, Subcode 1

Issue the .GCMAP request to return the CMAP status. The value is returned in R0.

.GCMAP area,CODE=strg

where:

area	is the address of a two-word EMT request block area
CODE=strg	specifies *strg* as either "SET" (default), "NOSET", "SP" or "STACK"

Errors:
None.

.GFDAT

EMT 375, Code 44

The .GFDAT programmed request returns in R0 the creation date from a file's directory entry.

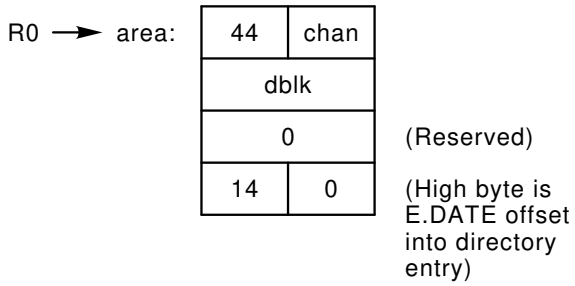
.GFDAT area,chan,dblk

where:

area is the address of a 4-word EMT argument block

chan is a channel number.

dblk is the address of a 4-word block containing a device and file specification in Radix–50.



Errors:

Code	Explanation
0	Channel is not available
1	File not found, or not a file-structured device.
2	Invalid operation
3	Invalid EMT request block

.GFINF

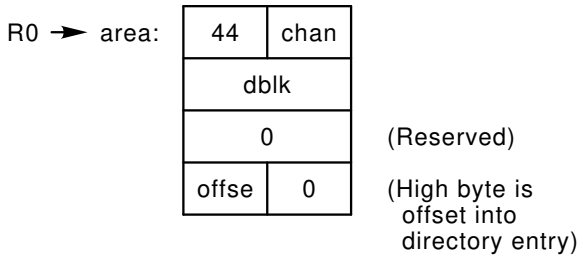
EMT 375, Code 44

The .GFINF programmed request returns in R0 the contents of the directory entry offset you specify.

.GFINF area,chan,dblk,offse

where:

area	is the address of a 4-word EMT argument block
chan	is a channel number.
dblk	is the address of a 4-word block containing a device and file specification in Radix–50.
offse	is the octal byte offset for the directory entry word you want. The offset must be even.



Errors:

Code	Explanation
0	Channel is not available
1	File not found, or not a file-structured device.
2	Invalid operation
3	Invalid offset value

.GFSTA

EMT 375, Code 44

The .GFSTA programmed request returns in R0 the status information from a file's directory entry (E.STAT word).

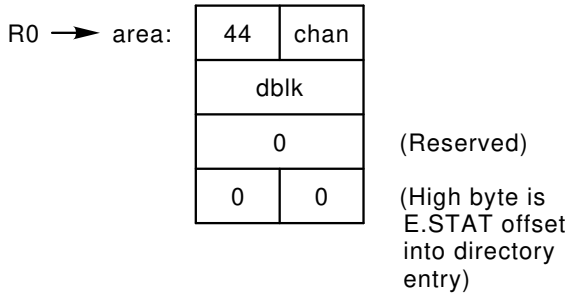
.GFSTA area,chan,dblk

where:

area is the address of a 4-word EMT argument block

chan is a channel number.

dblk is the address of a 4-word block containing a device and file specification in Radix–50.



Errors:

Code	Explanation
0	Channel is not available
1	File not found, or not a file-structured device.
2	Invalid operation
3	Invalid offset value

.GMCX

Mapping Only

EMT 375, Code 36, Subcode 6

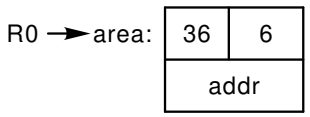
The GMCX request returns the mapping status of a specified window in the window definition block.

.GMCX area, [,addr]

where:

area is the address of a two-word EMT argument block

addr is the address of the window definition block where the specified window's status is returned



The .GMCX request modifies the following fields of the window definition block:

W.NAPR base page address register of the window

W.NBAS window virtual address

W.NSIZ window size in 32-word blocks

W.RID region identifier

If the window whose status is requested is mapped to a region, the .GMCX request loads the following additional fields in the window definition block; otherwise, these locations are zeroed:

W.NOFF offset value into the region

W.NLEN length of the mapped window

W.NSTS state of the WS.MAP bit is set to 1 in the window status word

Errors:

Code	Explanation
3	An invalid window identifier was specified
17	Inactive space or mode was specified

.GTIM

EMT 375, Code 21

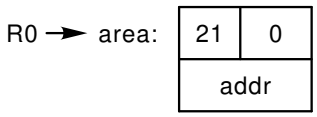
.GTIM accesses the current time of day. The time is returned in two words and given in terms of clock ticks past midnight.

.GTIM area, addr

where:

area is the address of a two-word EMT argument block

addr is the address of the two-word area where the time is to be returned



The high-order time is returned in the first word; the low-order time is returned in the second word. Your program must perform the conversion from clock ticks to hours, minutes, and seconds.

Errors:
None.

.GTJB

EMT 375, Code 20, Subcode 1

The .GTJB request returns information about a job in the system.

.GTJB area,addr[,jobblk]

where:

area is the address of a three-word EMT argument block

addr is the address of an eight-word or twelve-word block reply area. (The *addr* values are listed below.)

jobblk is a pointer to a three-word ASCII logical job name for which data are being requested. If the *jobblk* argument to the .GTJB request is:

- Between 0 and 16, it is interpreted as a job number.
- 'ME' or equals -1, the current job.
- Omitted or equals -3 only eight words of information are returned.

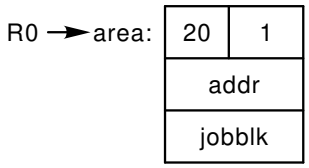
Addr Values:

Word Offset	Contents
Word 0 J.BNUM	Job Number: • System Job Monitors: Background job is 0 System jobs are 2–14 Foreground job is 16 • Non-System Monitors: Background job is 0 Foreground job is 2 • Single Job Monitor: Job number is 0
Word 1 J.BHLM	High-memory limit of job partition (highest location available to a job in low memory if the job executes a privileged .SETTOP -2 request)

Word 2 J.BLLM	Low-memory limit of job partition (first location)
Word 3 J.BCHN	Pointer to I/O channel space
Word 4 J.BIMP	Address of job's impure area in monitors
Word 5 J.BLUN	Low byte: unit number of job's console terminal; 0 when multiterminal feature is not used. High byte: reserved for future use
Word 6 J.BVHI	Virtual high limit for a job created with the linker /V option (XM only; 0 when not running under the mapped monitor or if /V option is not used)
Word 7–8	Reserved for future use
Word 9–11 J.BLNM	ASCII logical job name (System Job Monitors only; contains nulls for non-system job monitors.)

Word 3 normally indicates an address within the job's impure area. However, when a .CDFN is executed, the start of the I/O channel area changes to the user-specified area.

In an environment without the system job feature, you can get another job's status only by specifying its job number (0 or 2).



Errors:

Code	Explanation
0	No such job currently running.

.GTLIN

EMT 345

The .GTLIN request collects a line of input from the terminal or an indirect command file, if one is active.

.GTLIN linbuf[,prompt][,type]

where:

linbuf is the address of the buffer to receive the input line. This area must be at least 81 bytes in length. The input line is stored in this area and is terminated with a zero byte

prompt is the address of a prompt string to be printed on the console terminal. Usually, the string ends with an octal 200 byte to suppress printing the RETURN at the end of the prompt

type is an optional argument which, if specified, forces .GTLIN to take its input from the terminal rather than from an indirect file.

Errors:

Code	Explanation
0	Invalid command line (if the line is too long)

.GVAL

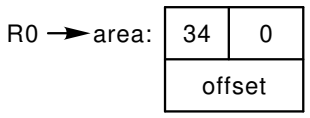
.GVAL: EMT 375, Code 34, Subcode 0

The .GVAL request returns in R0 the contents of a monitor fixed offset.

.GVAL area, offset

area is the address of a two-word EMT argument block

offset is the displacement from the beginning of the resident monitor to the word to be returned in R0



Errors:

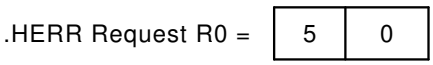
Code	Explanation
0	The offset requested is beyond the limits of the resident monitor.
1	Odd address.

.HERR

.HERR: EMT 374, Code 5

.HERR turns off user error interception. It allows the system to abort the job on fatal errors and generate an error message. (.HERR is the default case.)

.HERR



Errors:

The .HERR and .SERR programmed requests return a code value in R0 that lets a subroutine implicitly control error condition handling:

Code	Explanation
0	Previous state of error condition handling (.HERR/.SERR flag) was .HERR
1	Previous state of error condition handling (.HERR/.SERR flag) was .SERR

.HRESET

EMT 357

The .HRESET request stops all I/O transfers in progress for the issuing job, and then performs an .SRESET request.

.HRESET

Errors:
None.

.INTEN

Macro Expansion

Interrupt service routines use .INTEN:

- To notify the monitor that an interrupt has occurred and to switch to system state.
- To set the processor priority to the correct value.
- To save the contents of R4 and R5 before returning to the Interrupt Service Routine. Any other registers must be saved by the routine.

.INTEN prio[,pic]

where:

prio is the processor priority at which the interrupt routine is to run, normally the priority at which the device requests an interrupt

pic is an optional argument that should be non-blank if the interrupt routine is written as a PIC (position-independent code) routine. Any interrupt routine written as a device handler must be a PIC routine and must specify this argument

Errors:
None.

.LOCK

EMT 346

The .LOCK request keeps the USR in memory to provide any of its services required by your program. A .LOCK inhibits another job from using the USR.

.LOCK

Errors:
None.

.LOOKUP

EMT 375, Code 1

A .LOOKUP request can be used in two different ways:

- As a standard .LOOKUP file under all monitors.
- As an MQ job .LOOKUP under system job monitors.

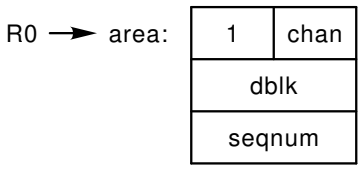
Standard .LOOKUP

.LOOKUP area,chan,dblk[,seqnum]

where:

area	is the address of a three-word EMT argument block
chan	is a channel number in the range 0-376_8
dblk	is the address of a four-word Radix–50 descriptor of the file or device to be operated upon
seqnum	is a file number for magtape.

On return from the .LOOKUP, R0 contains the length in blocks of the file just opened. On a return from a .LOOKUP for a non-directory, file-structured device (typically magtape), R0 contains 0 to indicate the unknown length.



Errors:

Code	Explanation
0	Channel already open.
1	File indicated was not found on the device.

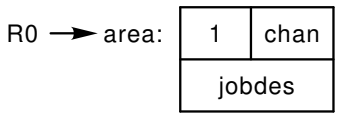
2	File already open on a nonshareable device; for example, magtape.
5	Argument is invalid; for example, magtape file sequence number.
6	Error code is returned in $ERRBY if the request is issued to a nonexistent or otherwise invalid special-directory device unit. The handler determines the validity of the device unit.

System Job .LOOKUP

.LOOKUP area,chan,jobdes

where:

area	is the address of a two-word EMT argument block
chan	is the number of the channel to open
jobdes	is the address of a four-word descriptor of the job to which messages will be sent or received. A *logical-job-name* can be from one to six characters long. It must be padded with nulls if less than six characters long. If a *logical-job-name* is all nulls, the channel will be opened for .READ requests only and will accept messages from any job.



The .LOOKUP request associates a channel with a specified job for the purposes of sending inter-task messages. R0 is undefined on return from the .LOOKUP.

Errors:

Code	Explanation
0	Channel not available.
1	No such job.

.MAP

Mapping Only

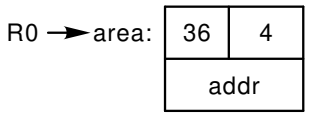
.MAP: EMT 375, Code 36, Subcode 4

The .MAP request maps a previously defined address window.

.MAP area[,addr]

where:

area	is the address of a two-word EMT argument block
addr	is the address of the window definition block



Errors:

Code	Explanation
2	An invalid region identifier was specified.
3	An invalid window identifier was specified.
4	The specified window was not mapped because the offset is beyond the end of the region, the window is larger than the region or extends beyond the bounds of the region.

.MFPS

Macro Expansion

The .MFPS call allows processor-independent user access to the processor status word. The contents of the registers are preserved across either call. The .MFPS call reads the priority bits only. Condition codes are destroyed during the call and must be directly accessed (using conditional branch instructions) if they are to be read in a processor-independent manner.

.MFPS addr

where:

addr is the address into which the processor status is to be stored; if addr is not present, the value is returned on the stack. Note that only the priority bits are significant

Errors:
None.

.MRKT

Timer

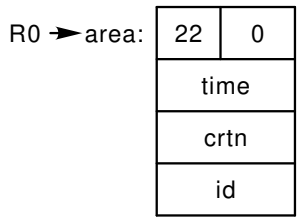
EMT 375, Code 22

The .MRKT request schedules a completion routine to be entered after a specified time interval has elapsed.

.MRKT area,time,crtn,id

where:

area	is the address of a four-word EMT argument block
time	is the address of a two word-block containing the time interval (high order first, low order second), specified as a number of clock ticks
crtn	is the entry point of a completion routine
id	is a non-zero number or memory address used to identify the particular request to the completion routine. The number must be within the range 1–176777; the rest are reserved for system use. The number need not be unique. On entry to the completion routine, the *id* number is in R0



Errors:

Code	Explanation
0	No queue element was available.

.MSDS

Fully Mapped

EMT 375, Code 46, Subcode 0

The .MSDS sets the lockstep of User data space and Supervisor data space and returns the old status in R0. This value is not implemented in unmapped monitors.

.MSDS area,value[,CODE=strg]

where:

area	is the address of a two-word EMT request block area
value	is the setting value desired.
CODE=strg	specifies *strg* as either "SET" (default), "NOSET", "SP" or "STACK"

Errors:
None.

.MTATCH

Multiterminal

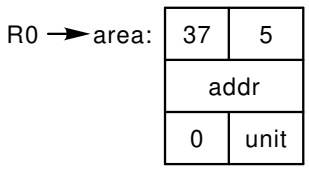
EMT 375, Code 37, Subcode 5

The .MTATCH request attaches a terminal for exclusive use by the requesting job.

.MTATCH area,addr,unit

where:

area	is the address of a three-word EMT argument block
addr	is the optional address of an asynchronous terminal status word, or it must be #0. (The asynchronous terminal status word is a system option you can select during the system generation process.)
unit	is the logical unit number of the terminal (The logical unit number is the number assigned by the system to a particular physical unit during the system generation process.)



Errors:

Code	Explanation
2	Nonexistent logical unit number.
3	Invalid request; function code out of range.
4	Unit attached by another job (job number returned in R0).
5	In mapped monitors, the optional status word address is not in valid user virtual address space.
6	Unit attached by a handler (Radix–50 handler name returned in R0)

.MTDTCH

Multiterminal

EMT 375, Code 37, Subcode 6

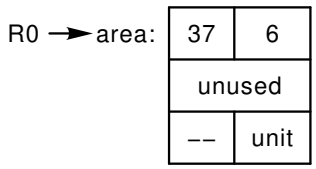
The request detaches a terminal from one job and makes it available for other jobs.

.MTDTCH area,unit

where:

area is the address of a three-word EMT argument block

unit is the logical unit number (lun) of the terminal to be detached



Errors:

Code	Explanation
1	Unit not attached.
2	Nonexistent logical unit number.
3	Function code out of range.

.MTGET

Multiterminal

EMT 375, Code 37, Subcode 1

.MTGET returns terminal status.

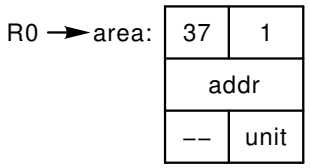
.MTGET area,addr,unit

where:

area is the address of a three-word EMT argument block.

addr is the address of a four-word status block where the status information is returned. If the low order bit is zero, the status block is in User space. If the low order bit is one, the status block is in Supervisor space.

unit is the logical unit number (*lun*) of the terminal whose status is requested. A unit need not be attached to the job issuing a .MTGET request. If the unit is attached to another job (error code 4), the terminal status will be returned and the job number will be contained in R0. For all other error conditions, the contents of R0 are undefined.



The status block has the following structure:

M.TSTS	
M.TST2	
M.FCNT	M.TFIL
M.TSTW	M.TWID

The following information is contained in the status block:

Byte Offset		Description
0	(M.TSTS)	Terminal configuration word 1
2	(M.TST2)	Terminal configuration word 2
4	(M.TFIL)	Character requiring fillers
5	(M.FCNT)	Number of fillers
6	(M.TWID)	Carriage width
7	(M.TSTW)	Terminal status byte (high byte of TSTDF)

Note that if an error occurs, and the error code is not 1 or 4, the status block will not have been modified.

Terminal Configuration Word 1 - M.TSTS

The bit definitions for terminal configuration word 1 (M.TSTS) are as follows:

Name	Value	Meaning
HWTAB$	1	Terminal has hardware tab
CRLF$	2	Output RETURN when carriage width exceeded
FORM$	4	Terminal has hardware form feed
FBTTY$	10	Process CTRL/F and CTRL/B (and CTRL/X if system job) as special command characters (If clear, CTRL/F and CTRL/B are treated as ordinary characters.)
TCBIT$	100	Inhibit TT wait (similar to bit 6 in the Job Status Word)
PAGE$	200	Enable CTRL/S-CTRL/Q processing
	7400	Line speed (baud rate) mask. The terminal baud rate values for DZ11/DZV11 and DH for bits 11–8 are as follows:

Mask (M.TSTS bits 11–8)	DZ Baud Rate	DH Baud Rate
0000	50	50
0400	75	75
1000	110	110
1400	134.5	134.5
2000	150	150
2400	300	300
3000	600	600
3400	1200	1200
4000	1800	1800
4400	2000	2000
5000	2400	2400
5400	3600	38400
6000	4800	4800
6400	7200	7200
7000	9600	9600
7400	n/a	19200

Name	Value	Meaning
TTSPC$	10000	Character mode input (same as bit 12 in Job Status Word)
REMOT$	20000	Terminal is remote (Read-only bit)
TTLC$	40000	Lowercase to uppercase conversion disabled
BKSP$	100000	Use backspace for rubout (video type display)

Terminal Configuration Word 2 - M.TST2

The bit definitions for terminal configuration word 2 (M.TST2) are as follows:

Name	Value	Meaning
CHRLN$	3	Character length, which can be 5(00), 6(01), 7(10), or 8(11) bits (DZ only)
USTOP$	4	Unit stop, which sends one stop bit when clear, two stop bits when set (DZ only)
PAREN$	10	Parity enable (DZ only)
ODDPR$	20	Odd parity when set; even parity when clear
	140	Reserved
RPALL$	200	Read pass all
	77400	Reserved
WPALL$	100000	Write pass all

Terminal Status Byte - M.TST2

The bit definitions for terminal status byte (M.TST2) are as follows:

Name	Value	Meaning
FILL$	1	Fill sequence in progress
CTRLU$	2	CTRL/U in progress
DTACH$	20	Detach in progress
WRWT$	40	TT: I/O flag
INEXP$	100	Output interrupt is expected
PAGE$	200	Output is suspended by XOFF
SHARE$	2000	Terminal is shared console
HNGUP$	4000	Terminal has hung up
DZ11$	10000	Terminal interface is DZ11
CTRLC$	40000	Double CTRL/C was entered
CONSL$	100000	Terminal is acting as console

Errors:

Code	Explanation
1	Invalid unit number, unit not attached.
2	Nonexistent logical unit number.
3	Invalid request; function code out of range.
4	Unit attached by another job (job number returned in R0).
5	In the XM monitor, the status block address is not in valid user virtual address space.
6	Unit attached by a handler (Radix–50 handler name returned in R0)

.MTIN

Multiterminal

EMT 375, Code 37, Subcode 2

The .MTIN request moves one or more characters from the input ring buffer to a buffer you specify. The terminal must be attached. An updated user buffer address is returned in R0 if the request is successful. The .MTIN request has the following form:

.MTIN area,addr,unit[,chrcnt]

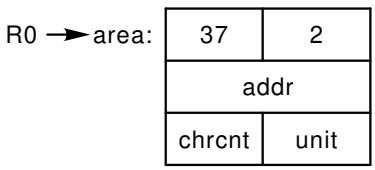
where:

area is the address of a three-word EMT argument block.

addr is the byte address of the input buffer.

unit is the logical unit number.

chrcnt is a character count indicating the number of characters to transfer. The valid range is from 0 to 255(decimal). A character count of zero means one character



Errors:

Code	Explanation
0	No input available. TCBIT$ is set in the Job Status Word (for the system console) or in M.TSTS by the .MTSET request.
1	Unit not attached.
2	Nonexistent logical unit number.
3	Function code out of range.
5	In the mapped monitor, the user buffer address is not in valid user virtual address space.

.MTOUT

Multiterminal

EMT 375, Code 37, Subcode 3

The .MTOUT request places characters into the user buffer.

.MTOUT area,addr,unit[,chrcnt]

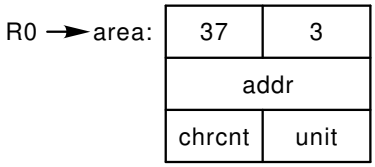
where:

area is the address of a three-word EMT argument block.

addr is the address of the output buffer.
If low order bit of *addr* word is zero, the data buffer is in User space.
If low order bit is one, the data buffer is in Supervisor space.

unit is the unit number of the terminal

chrcnt is a character count indicating the number of characters to transfer.
The valid range is from 1 to 255(decimal)



Errors:

Code	Explanation
0	No room in output buffer.
1	Unit not attached.
2	Nonexistent logical unit number.
3	Function code out of range.
5	In the mapped monitor, the user buffer address is not in valid user virtual address space.

.MTPRNT

Multiterminal

EMT 375, Code 37, Subcode 4

The .MTPRNT request causes one or more lines to be printed at the specified terminal in a multiterminal environment.

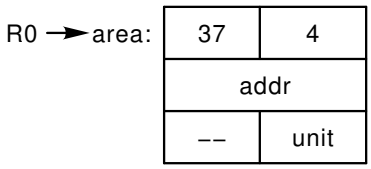
.MTPRNT area,addr,unit

where:

area is the address of a three-word EMT argument block

addr is the starting address of the character string to be printed. If the low order bit of the *addr* word is zero, the data buffer is in User space; if the low order bit is one, the data buffer is in Supervisor space.

unit is the unit number associated with the terminal



Errors:

Code	Explanation
1	Unit not attached.
2	Nonexistent logical unit number.
3	Function code out of range.
5	In the mapped monitor, the character string address is not in valid user virtual address space.

.MTPS

Macro Expansion

The .MTPS macro call allows processor-independent user access to the processor status word. The contents of the registers are preserved across either call. The .MTPS call sets the priority bits.

.MTPS value

where:

value is either the value or the address of the value (depending on addressing mode) to be placed in the PSW. If value is not present, the processor status word is taken from the stack. Note that the high byte on the stack is set to 0 when value is present. If value is not present, you should set the stack to the appropriate value. In either case, the lower byte on the stack is put in the processor status word.

Errors:
None.

.MTRCTO

Multiterminal

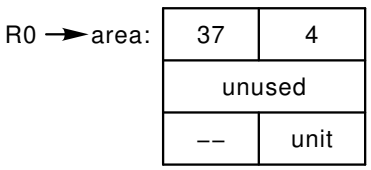
EMT 375, Code 37, Subcode 4

The .MTRCTO request resets the CTRL/O switch of the specified terminal and enables terminal output in a multiterminal environment.

.MTRCTO area,unit

where:

area	is the address of a three-word EMT argument block
unit	is the unit number associated with the terminal



Errors:

Code	Explanation
1	Unit not attached.
2	Nonexistent logical unit number.
3	Function code out of range.

.MTSET

Multiterminal

EMT 375, Code 37, Subcode 0

.MTSET is a multiterminal feature that:

- Sets terminal and line characteristics.
- Determines the input/output mode of the terminal service requests for the specified terminal.

For more detail on line characteristics, such as baud rate, number of data bits, stop bits, and parity, refer to .MTGET.

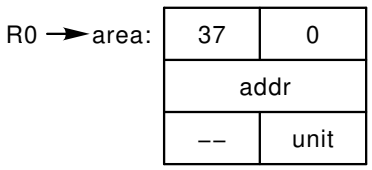
.MTSET area,addr,unit

where:

area is the address of a three-word EMT argument block

addr is the address of a four-word status block containing the line and terminal status being set.
If low order bit of *addr* word is zero, the status block is in User space.
If low order bit is one, the status block is in Supervisor space.

unit is the logical unit number associated with the line and terminal



.MTSTAT

EMT 375, Code 37

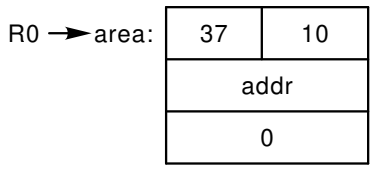
The .MTSTAT request returns multiterminal system status information.

.MTSTAT area,addr

where:

area	is the address of a three-word EMT block.
addr	is the address of an eight-word return block.

Byte Offset	Explanation
0 (MST.1T)	Offset from the base of the resident monitor to the first terminal control block (TCB)
2 (MST.CT)	Offset from the base of the resident monitor to the terminal control block of the console terminal for the program
4 (MST.LU)	The value (0-16 decimal) of the highest LUN
6 (MST.ST)	The size of the terminal control block in bytes
10-17	Reserved



Errors:

Code	Explanation
3	Function code out of range
5	In mapped monitors, the status block address is not in valid user address space.

.MWAIT

Multijob

EMT 374, Code 11, Subcode 0

This request is similar to the .WAIT request, except that .MWAIT suspends execution of the job issuing the request until all messages sent to the other job or requested from the other job have been received. Using .MWAIT with .RCVD or .SDAT modes of message handling causes the program to wait until data is transferred. This request is available only under multijob monitors.

.MWAIT

where:

R0 =

11	0

Errors:
None.

.PEEK

EMT 375, Code 34, Subcode 1

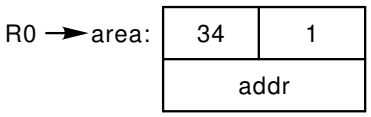
The .PEEK programmed request accesses processor status and returns in R0 the contents of a specified low memory location (below 28K words) or I/O page location. The request can access the processor status (PS) word as ADDR 177776.

.PEEK area,addr

where:

area is the address of a two- or three-word EMT argument block

addr is the address of the location to examine or change



Errors:

Code	Explanation
1	Odd or nonexistent address.

.POKE

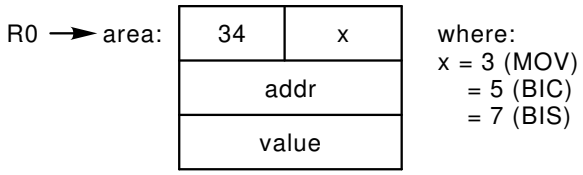
EMT 375, Code 34, Subcode 3, 5, 7

.POKE deposits the value you specify into a low memory location (below 28K words) or I/O page location and returns the old contents of the memory location in R0. This simplifies the saving and restoring of a location. .POKE supports BIC (bit clear) and BIS (bit set) operations, as well as the previous MOV operation, using an optional parameter, *type*.

.POKE area,addr,value[,type]

where:

area	is the address of a three-word EMT argument block
addr	is the address of the location to examine and change
value	is the new contents to place in the location
type	is the instruction used to modify the address to the specified value. The *type* parameter can be BIC, BIS, or MOV (default).



Errors:

Code	Explanation
1	Odd or nonexistent address

The .POKE request can alter priority bits in the processor status (PS) word:

- A .POKE request returns the contents of the PS with undefined condition codes.
- .POKE can modify all bits in the PS except the 000020 (trace trap) and 140000 (current mode) bits. However, modifying the 000400 (instruction suspension) or 004000 (register set) bits can cause unexpected results. Although not prohibited, this is not recommended.

Modifying priority bits is supported. However, changing processor priority with a .POKE request automatically lowers processor priority to zero (PR0) during the time .POKE is executing. Therefore, a period of lowest processor priority exists between the time the processor is running at a given priority and the time the processor priority change takes effect.

Setting the carry bit is not recommended, as it causes .POKE to return an error.

- The priority portion of the modified PS is preserved at the completion of the .POKE request.

.PRINT

EMT 351

The .PRINT request causes output to be printed at the console terminal.

.PRINT addr

where:

addr is the address of the string to be printed

The string to be printed can be terminated with either a null (0) byte or a 200 byte. If the null (ASCIZ) format is used, the output is automatically followed by a RETURN/LINE FEED combination. If a 200 byte terminates the string, no RETURN/LINE FEED combination is generated.

NOTE

Unlike other SYSMAC.MAC definitions, .ENABL MCL will not work for .PRINT; therefore, to use .PRINT, you must do a .MCALL for it.

Errors:
None.

.PROTECT

EMT 375, Code 31, Subcode 0

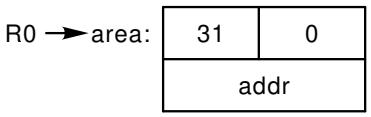
The .PROTECT request allows a job to obtain exclusive control of a vector (two words) in the area of 0 to 474.

.PROTECT area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of the word pair to be protected



Errors:

Code	Explanation
0	Protect failure; locations already in use.
1	Address *addr* is greater than 474 or is not a multiple of 4.

.PURGE

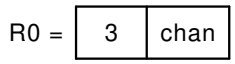
EMT 374, Code 3

The .PURGE request makes a channel available. If a tentative file has been entered on the channel, the file is discarded.

.PURGE chan

where:

chan is the number of the channel to be made available



Errors:
None.

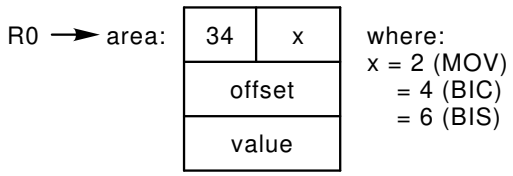
.PVAL

.GVAL: EMT 375, Code 34, Subcode 2

The .PVAL request returns the old contents of an offset in R0 to simplify saving and restoring an offset value.

.PVAL area,offset,value[,TYPE=strg]

area	is the address of a three-word EMT argument block
offset	is the displacement from the beginning of the resident monitor to the word to be returned in R0
value	is the new value to be placed in the fixed offset location
type	is the instruction used to modify the address to the specified value. The *type* parameter can be BIC, BIS, or MOV (default). BIS uses *value* to set *offset*.



Errors:

Code	Explanation
0	The offset requested is beyond the limits of the resident monitor.
1	Odd address.

.QELDF

Macro Expansion

The .QELDF macro symbolically defines queue element offsets. Since the handler usually deals with queue element offsets relative to Q.BLKN, the .QELDF macro also defines these associated symbolic offsets, using a $ character substituted for the period (.).

QDELF list, E

In the following example, if memory address conditional (MMG$T) equals 1 (default), additional offsets are generated for use only for mapped monitors. The length of queue element (Q.ELGH) is controlled by the MMG$T setting.

```
Normal Offset   Handler Offset          Description
-------------   --------------          ------------
Q.LINK=0        Q$LINK=Q.LINK-^o4       Link to next queue element
Q.CSW=2.        Q$CSW=Q.CSW-^o4         Pointer to channel status word
Q.BLKN=4.       Q$BLKN=Q.BLKN-^o4       Physical block number
Q.FUNC=6.       Q$FUNC=Q.FUNC-^o4       Special function code
Q.2UNI=Q.FUNC   Q$2UNI=Q.2UNI-^o4       High 3 bits of unit number(3)
Q.TYPE=Q.FUNC   Q$TYPE=Q.TYPE-^o4       Normal I/O or special function flag(3)
Q.FMSK=^o17                             Special function bits mask(3)
Q.2MSK=^o160                            High 3 bits of unit number mask(3)
Q.TMSK=^o200                            Normal or special flag mask(3)
Q.JNUM=7.       Q$JNUM=Q.JNUM-^o4       Job number
Q.UNIT=Q.JNUM   Q$UNIT=Q.UNIT-^o4       Low (or all) bits of unit number
Q.UMSK=^o3400                           Unit number mask
Q.JMSK=^o74000                          Job number mask
Q.BUFF=^o10     Q$BUFF=Q.BUFF-^o4       Buffer physical address
Q.WCNT=^o12     Q$WCNT=Q.WCNT-^o4       Transfer count
Q.COMP=^o14     Q$COMP=Q.COMP-^o4       Completion routine or I/O type flag

Q.PAR=^o16(2)   Q$PAR=Q.PAR-^o4(2)      DMA PAR1 base address
Q.MEM=^o20(2)   Q$MEM=Q.MEM-^o4(2)      CPU access PAR1 base address

Q.ELGH=^o16(1)                          Length of queue element
Q.ELGH=^o24(2)                          Length of queue element

(1) unmapped system (MMG$T=0)
(2) mapped system (MMG$T=1)
(3) extended unit handlers and systems only
```

.QSET

EMT 353

.QSET request enables additional entries to the RT–11 I/O free queue.

.QSET addr,len

where:

addr is the address at which the new elements are to start

len is the number of entries to be added. In the unmapped monitors, each queue entry is seven words long; hence the space set aside for the queue should be *len*7* words. In the mapped monitors, 10(decimal) words per queue element are required

Errors:

In an extended memory environment, an attempt to violate the PAR1 restriction results in a *?MON-F-addr* error, which can be intercepted with a .SERR programmed request.

.RCTRLO

EMT 355

The .RCTRLO request resets the CTRL/O flag for the terminal.

.RCTRLO

Errors:

If the terminal is set to XOFF (no scroll or hold session), .RCTRLO will not clear the XOFF condition.

.RCVD

Multijob

EMT 375, Code 26

The .RCVD (receive data) request allows a job to read messages or data sent by another job in a multijob environment.

The request is posted and the issuing job continues execution. When the job needs to have the transmitted message, executing .MWAIT suspends the job until all .SDATx and .RCVDx requests for the job are complete.

.RCVD area,buf,wcnt[,BMODE=strg]

where:

area is the address of a five-word EMT argument block

buf is the address of the buffer into which the message length and message data are to be placed

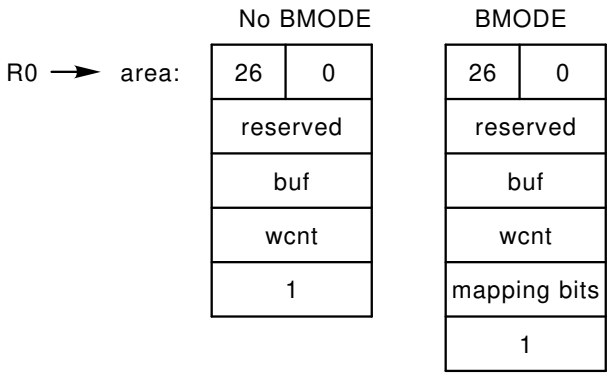
wcnt is the number of words in the buffer

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	No other job exists in the system. (A job exists as long as it is loaded, whether or not it is active.)

.RCVDC

Multijob

EMT 375, Code 26

The .RCVDC request receives data and enters a completion routine when the message is received. The .RCVDC request is posted and the issuing job continues to execute. When the other job sends a message, the completion routine specified is entered.

.RCVDC area,buf,wcnt,crtn[,BMODE=strg][,CMODE=strg]

where:

area is the address of a five-word EMT argument block

buf is the address of the buffer into which the message length /message data is to be placed

wcnt is the number of words in the buffer

crtn is the address of a completion routine to be entered

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.

CMODE=strg where *strg* is:

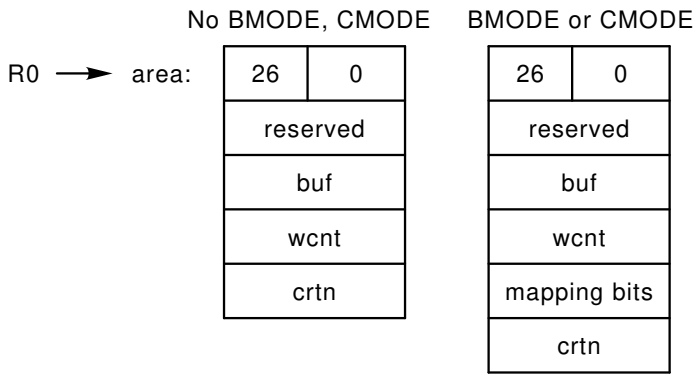
Value	Description
U	User space (default)
S	Supervisor space

Specifying *CMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *CMODE*.
- Specifies the *space* for the *crtn* argument.
- Is a valid option only for fully mapped monitors.

Note that, if both BMODE and CMODE are specified, only one additional word is added.

As in the .RCVD request, word 0 of the buffer contains the number of words transmitted when the transfer is complete.



Errors:

Code	Explanation
0	No other job exists in the system. (A job exists as long as it is loaded, whether or not it is active.)

.RCVDW

Multijob

EMT 375, Code 26

A message request is posted and the job issuing the request is suspended until all pending .SDATx and .RCVDx requests for the job are complete. When the issuing job runs again, the message has been received, and word 0 of the buffer indicates the number of words transmitted.

.RCVDW area,buf,wcnt[,BMODE=strg]

where:

area is the address of a five-word EMT argument block

buf is the address of the buffer into which the message length /message data is to be placed

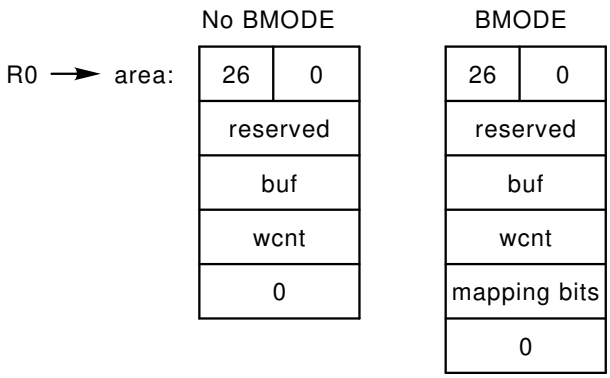
wcnt is the number of words to be transmitted

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	No other job exists in the system. (A job exists as long as it is loaded, whether or not it is active.)

.RDBBK

Mapping

Macro Expansion

The .RDBBK macro defines symbols for the region definition block and reserves space for it. The .RDBBK automatically invokes .RDBDF.

.RDBBK rgsiz,rgsta,name[,BASE=n]

where:

rgsiz	is the size of the dynamic region needed (expressed in 32-word units)
rgsta	is the region status byte
name	is the name of the global region
BASE=n	specifies the starting address of the region, expressed in 32-word units. A value of 0 (or value omitted) means any available base address is acceptable.

Errors:
None.

.RDBDF

Mapping

Macro Expansion

The .RDBDF macro defines the symbolic offset names for the region definition block and the names for the region status word bit patterns. This macro also defines the length of the region definition block, but it does not reserve space for the region definition block.

.RDBDF

The .RDBDF macro expands as follows:

```
R.GID =: 0              ;Region ID
R.GSIZ=: 2.             ;Size in chunks
R.GSTS=: 4.             ;Status
R.GLLN=: 6.             ;Length of simple RDB
R.GNAM=: 6.             ;Name of global region (optional)
R.GBAS=: 10.            ;Base physical address (optional)
R.GLGH=: 12.            ;Length of complex RDB
RS.CRR=: ^o100000       ;Region created successfully
RS.UNM=: ^o40000        ;One or more windows eliminated
RS.NAL=: ^o20000        ;Region newly allocated
RS.NEW=: ^o10000        ;New global region
RS.GBL=: ^o4000         ;Create within a global region
RS.CGR=: ^o2000         ;Create new global region if not found
RS.AGE=: ^o1000         ;Enable auto elimination of region
RS.EGR=: ^o400          ;Eliminate global region
RS.EXI=: ^o200          ;Eliminate global region on exit / abort
RS.CAC=: ^o100          ;Bypass memory cache
RS.BAS=: ^o40           ;R.GBAS supplied
RS.NSM=: ^o20           ;Non-system memory
RS.DSP=: ^o2            ;reserved
RS.PVT=: ^o1            ;reserved
```

.READ

EMT 375, Code 10

The .READ request transfers to memory a specified number of words from the device associated with the specified channel. Control returns to the user program immediately after the .READ is initiated, possibly before the transfer is completed. No special action is taken by the monitor when the transfer is completed.

.READ area,chan,buf,wcnt,blk[,BMODE=strg]

where:

area is the address of a five- or six-word EMT argument block.

chan is a channel number.

buf is the address of the buffer.

wcnt is the number of words to be read.

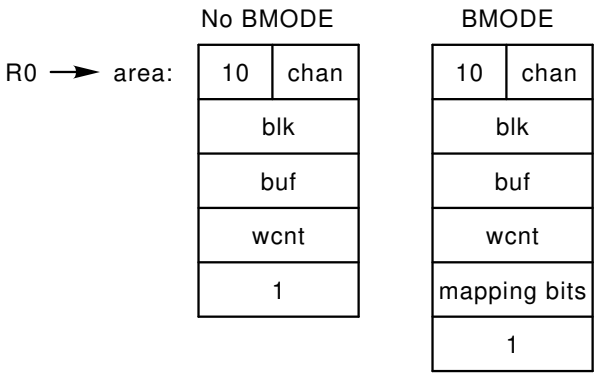
blk is the block number to be read. For a file-structured .LOOKUP, the block number is relative to the start of the file. For a non-file-structured .LOOKUP, the block number is the absolute block number on the device. The user program normally updates *blk* before it is used again. If input is from TT: and *blk=0*, TT: issues an up-arrow (^) prompt.

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



When the user program needs to access the data read on the specified channel, issue a .WAIT request as a check that the data has been read completely. If an error occurred during the transfer, the .WAIT request indicates the error.

Errors:

Code	Explanation
0	Attempt to read past end-of-file.
1	Hard error occurred on channel.
2	Channel is not open.

.READC

EMT 375, Code 10

The .READC request transfers to memory a specified number of words from the device associated with the specified channel. Control returns to the user program immediately after the .READC is initiated. Attempting to read past end-of-file also causes an immediate return, in this case with the carry bit set and the error byte set to 0. Execution of the user program continues until the .READC is complete, then control passes to the routine specified in the request. When a RETURN is executed in the completion routine, control returns to the user program.

.READC area,chan,buf,wcnt,crtn,blk[,BMODE][,CMODE]

where:

area	is the address of a five-word EMT argument block
chan	is a channel number in the range $0\text{-}376_8$
buf	is the address of the buffer to receive the data read
wcnt	is the number of words to be read
crtn	is the address of the user's completion routine. The address of the completion routine must be above 500_8
blk	is the block number to be read. For a file-structured .LOOKUP, the block number is relative to the start of the file. For a non-file-structured .LOOKUP, the block number is the absolute block number on the device. The user program normally updates *blk* before it is used again

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

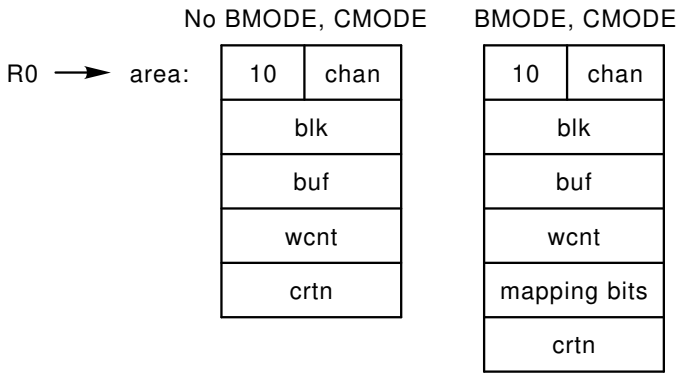
- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.

CMODE=strg where *strg* is:

Value	Description
U	User space (default)
S	Supervisor space

Specifying *CMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *CMODE*.
- Specifies the *space* for the *crtn* argument.
- Is a valid option only for fully mapped monitors.



Errors:

Code	Explanation
0	Attempt to read past end-of-file; no data was read.
1	Hard error occurred on channel.
2	Channel is not open.

.READW

EMT 375, Code 10

The .READW request transfers to memory a specified number of words from the device associated with the specified channel. When the .READW is complete or an error is detected, control returns to the user program.

.READW area,chan,buf,wcnt,blk[,BMODE=strg]

where:

area is the address of a five-word EMT argument block

chan is a channel number in the range $0\text{-}376_8$

buf is the address of the buffer to receive the data read

wcnt is the number of words to be read; each .READ request can transfer a maximum of 32K words

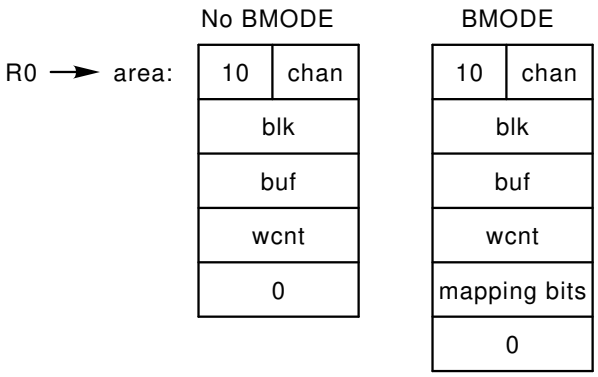
blk is the block number to be read. For a file-structured .LOOKUP, the block number is relative to the start of the file. For a non-file-structured .LOOKUP, the block number is the absolute block number on the device. The user program normally updates blk before it is used again

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	Attempt to read past end-of-file.
1	Hard error occurred on channel.
2	Channel is not open.

.RELEASE

EMT 343

The .RELEAS request notifies the monitor that a fetched device handler is no longer needed. .RELEAS of a valid device name from a foreground or system job is ignored.

.RELEAS dnam

where:

dnam is the address of the Radix–50 device name

Errors:

Code	Explanation
0	Device name is invalid.

.RENAME

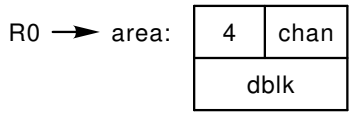
EMT 375, Code 4

The .RENAME request changes the name of the file specified.

.RENAME area,chan,dblk

where:

area	is the address of a two-word EMT argument block
chan	is an available channel number in the range $0\text{-}376_8$
dblk	is the address of a block that specifies the file to be renamed followed by the new file name



Errors:

Code	Explanation
0	Channel not available.
1	File not found.
2	Invalid operation.
3	A file by that name already exists and is protected. A RENAME was not done.

.REOPEN

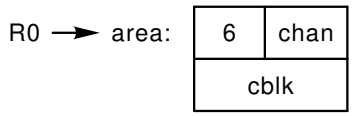
EMT 375, Code 6

The .REOPEN request associates the channel that was specified with a file on which a .SAVESTATUS was performed.

.REOPEN area,chan,cblk

where:

area	is the address of a two-word EMT argument block
chan	is a channel number in the range $0\text{-}376_8$
cblk	is the address of the five-word block where the channel status information was stored



Errors:

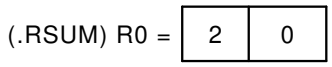
Code	Explanation
0	The specified channel is not available. The .REOPEN has not been done.

.RSUM

EMT 374, Code 2

The .RSUM request resumes control execution of a job's mainline code; that is, code that is not executing as a result of a completion routine. .RSUM (from a completion routine) resumes the mainline code.

.RSUM



Errors:
None.

.SAVESTATUS

EMT 375, Code 5

.SAVESTATUS stores, in a user-specified area of memory, the five-word channel status information RT–11 requires to completely define a file. .SAVESTATUS places data words in memory, frees the specified channel and closes the file. When the saved channel data is required, the .REOPEN request is used. The five words returned by .SAVESTATUS contain the following information:

Name	Offset	Contents
C.CSW	0	Channel status word
C.SBLK	2	Starting block number of this file or 0 if non-file-structured
C.LENG	4	Length of file
C.USED	6	Highest block written
C.DEVQ	10	Number of pending requests
C.UNIT	11	Device unit number

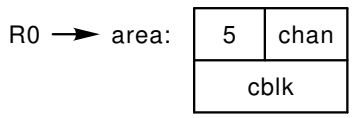
.SAVESTATUS area,chan,cblk

where:

area is the address of a two-word EMT argument block

chan is a channel number in the range 0-376_8

cblk is the address of the five-word user memory block where the channel status information is to be stored



.SAVESTATUS

Errors:

Code	Explanation
0	The channel specified is not open.
1	The file was opened with an .ENTER request or a .SAVESTATUS request was performed for a magtape file.

.SCCA

EMT 375, Code 35, Subcode 0, 1

The .SCCA programmed request:

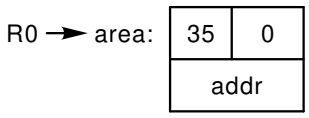
- Inhibits a CTRL/C abort
- Indicates when a double CTRL/C is initiated at the keyboard
- Distinguishes between single and double CTRL/C commands

.SCCA area,addr[,TYPE=strg]

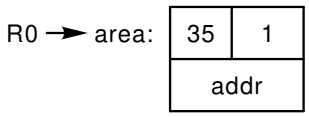
where:

area	is the address of a two-word parameter block
addr	is the address of a terminal status word (an address of 0 reenables CTRL/C aborts)

LOCAL:



GLOBAL:



Errors:
None.

.SDAT

Multijob

EMT 375, Code 25

The .SDAT/.SDATC/.SDATW requests are used with the .RCVD/.RCVDW/.RCVDC calls to allow message transfers between a foreground job and a background job under multijob monitors. Since RT–11 distinguishes a send from a receive by complementing the word count, a .SDAT* of zero words is treated as a .RCVDW of zero words. Therefore, avoid a word count of zero at all times when using a .SDAT* request.

.SDAT area,buf,wcnt[,BMODE=strg]

where:

area is the address of a five- or six-word EMT argument block

buf is the buffer address of the beginning of the message to be transferred

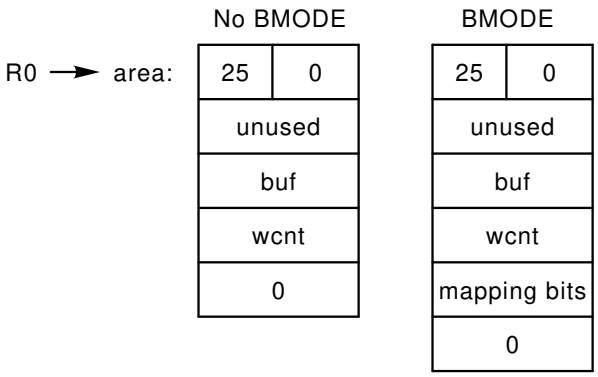
wcnt is the number of words to transfer

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	No other job exists. (A job exists as long as it is loaded, whether or not it is active.)

.SDATC

EMT 375, Code 25

.SDATC area,buf,wcnt,crtn[,BMODE=strg][,CMODE=strg]

where:

area is the address of a five-word or six-word EMT argument block

buf is the buffer address of the beginning of the message to be transferred

wcnt is the number of words to transfer

crtn is the address of the completion routine to be entered when the message has been transmitted

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

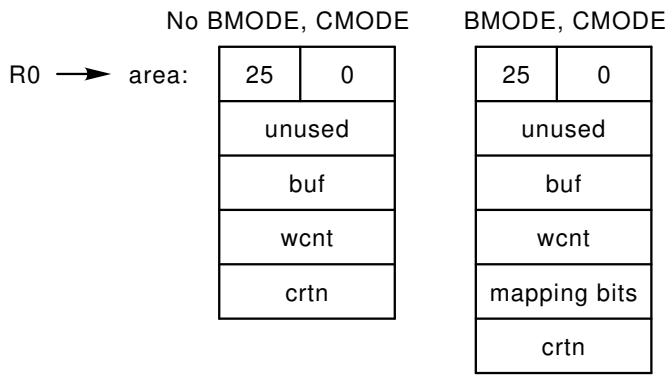
- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.

CMODE=strg where *strg* is:

Value	Description
U	User space (default)
S	Supervisor space

Specifying *CMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *CMODE*.
- Specifies the *space* for the *crtn* argument.
- Is a valid option only for fully mapped monitors.



Errors:

Code	Explanation
0	No other job exists. (A job exists as long as it is loaded, whether or not it is active.)

.SDATW

Multijob

EMT 375, Code 25

.SDATW area,buf,wcnt[,BMODE=strg]

where:

area is the address of a five- or six-word EMT argument block

buf is the buffer address of the beginning of the message to be transferred

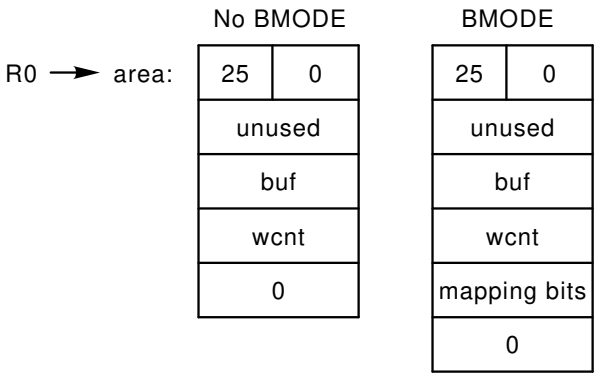
wcnt is the number of words to transfer

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	No other job exists. (A job exists as long as it is loaded, whether or not it is active.)

.SDTTM

EMT 375, Code 40

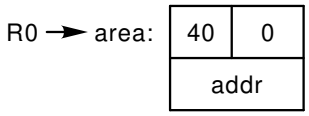
The .SDTTM (Set date and time) request allows your program to set the system date and time.

.SDTTM area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of a three-word block in user memory that contains the new date and time



Errors:
None.

.SERR

EMT 374, Code 4

.SERR inhibits the monitor from aborting a job and causing an error return to the offending EMT.

.SERR

where:

.SERR Request R0 =

4	0

Errors:

The .HERR and .SERR programmed requests return a code value in R0 that lets a subroutine implicitly control error condition handling:

Code	Explanation
0	Previous state of error condition handling (.HERR/.SERR flag) was .HERR
1	Previous state of error condition handling (.HERR/.SERR flag) was .SERR

Table 5–6: Soft Error Codes (.SERR)

Name	Code	Explanation
ER.USR	–1	USR. Reserved.
ER.UNL	–2	No device handler; this operation needs one.
ER.DIO	–3	Error doing directory I/O.
ER.FET	–4	.FETCH error. Either an I/O error occurred while the handler was being used, or an attempt was made to load the handler over USR or RMON.
ER.OVE	–5	Reserved. Issued by overlay handler, but not passed to a program.
ER.DFU	–6	No more room for files in the directory.

Table 5–6 (Cont.): Soft Error Codes (.SERR)

ER.IAD	–7	Invalid address. Tried to perform a monitor operation outside the job partition.
ER.ICH	–10	Invalid channel number; number is greater than actual number of channels that exist.
ER.EMT	–11	Invalid EMT; an invalid function code has been decoded.
	–12-13	Reserved for monitor internal use.
ER.DIR	–14	Invalid directory.
ER.XFE	–15	Unloaded handler under mapped system without fetch support.
	–16-22	Reserved for monitor internal use.

.SETTOP

EMT 354

The .SETTOP request specifies a new address as a program's upper limit. Using .SETTOP offers significant performance improvement in running your program.

.SETTOP addr

where:

addr is the address of the highest word of the area desired; that is, the last word the program will modify, not the first word it leaves untouched

On return from .SETTOP, both R0 and the word in $USRTO contain the highest memory address allocated for use.

Errors:
None.

.SFDAT

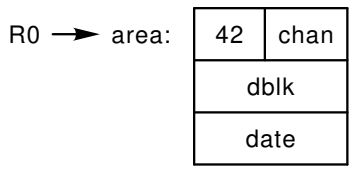
EMT 375, Code 42

The .SFDAT programmed request allows a program to set or modify the creation date in a file's directory entry. .SFDAT is not supported for distributed special directory handlers LP, LS, MM, MS, MT, MU, and SP.

.SFDAT area,chan,dblk[,date]

where:

area	is the address of a three-word EMT argument block.
chan	is a channel number.
dblk	is the address of a four-word block containing a filespec in Radix–50.
date	is the value of the new date, in RT–11 format. If this argument is #0 or omitted, the system date is used.



Errors:

Code	Explanation
0	Channel not available
1	File not found
2	Invalid operation (device not file structured)

.SFINF

EMT 375, Code 44

The .SFINF programmed request saves in R0 and then modifies the contents of the directory entry offset you specify from a file's directory entry. .SFINF is not supported for the distributed special directory handlers LP, LS, MM, MS, MT, MU, and SP.

.SFINF area,chan,dblk,value,type, offse,ucode

where:

area is the address of a 4-word or 5-word (see *dblk*) EMT argument block.

chan is a channel number.

dblk is a four-word Radix–50 descriptor block that specifies the physical device, file name, and file type to be operated upon.

value is the value used to modify the specified offset location.
For RT–11 file-structured volume directories,

- If the offset is 0 (E.STAT) and the operation is a BIC or BIS, E.STAT bits 000400, 001000, and 004000 must be clear.
- If the offset is E.STAT and the operation is a MOV, only the bottom 4 bits of E.STAT are moved.

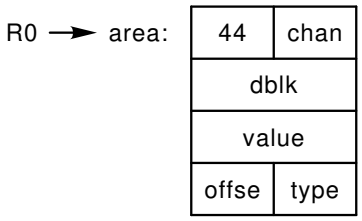
For special directory volumes, no bit restrictions are enforced. The operation is dependent on the handler.

type is the name indicating the operation to be performed:

Name	Value	Meaning
GET	0	Get value; a .GFDAT, .GFINF, or .GFSTA operation
BIC	1	A bit clear (BIC) operation
BIS	2	A bit set (BIS) operation
MOV	3	A word move (MOV) operation
	4-177	Reserved for Digital
USER	ucode	Reserved for the user

offse is the octal byte offset for the directory entry word for this operation. The offset must be even and cannot be 10 (E.LENG). For example, specifying offset 12 saves the current contents of E.USED in R0 and modifies that location according to the *value* and *type* arguments

ucode is the user operation code containing the value from the *type* parameter. Specify *ucode* only when the *type* parameter argument is USER



Errors:

Special directory handlers define their own codes. For RT–11 file-structured devices:

Code	Explanation
0	Channel is not available
1	File not found, or not a file-structured device.
2	Operation is invalid
3	Offset is invalid
4	Value is invalid

.SFPA

Floating-Point

EMT 375, Code 30

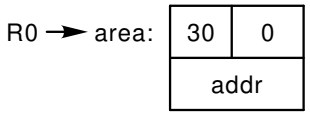
The .SFPA request enables users with floating-point hardware to set trap addresses to be entered when a floating-point exception occurs. Issue this request with #1 as the *addr* argument to provide swapping of floating-point context without setting a trap address. If no user trap address is specified and a floating-point (FP) exception occurs, a *?MON-F-FPU* trap occurs, and the job is aborted.

.SFPA area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of the routine to be entered when an exception occurs.
In a fully mapped monitor, if you set the low bit of *addr* on, it will be treated as Supervisor/Data space address; otherwise, it is treated as a User/Data Space address.



Errors:
None.

.SFSTAT

EMT 375, Code 44

The .SFSTAT programmed request saves in R0 and then modifies the contents of the directory entry E.STAT offset from a file's directory entry. .SFSTAT is not supported for the distributed special directory handlers LP, LS, MM, MS, MT, MU, and SP.

.SFSTAT area,chan,dblk,value,type,ucode

where:

area is the address of a 4-word (see *dblk*) EMT argument block

chan is a channel number.

dblk Four-word Radix–50 descriptor block that specifies the physical device, file name, and file type to be operated upon.

value is the value to be placed in the E.STAT offset location.
For RT–11 file-structured volume directories,

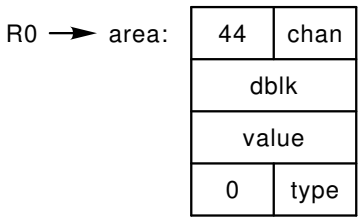
- If the operation is a BIC or BIS, E.STAT bits 000400, 001000, and 004000 must be clear.
- If the operation is a MOV, only the bottom 4 bits of E.STAT are moved.

For special directory volumes, no bit restrictions are enforced.

type is the name indicating the operation to be performed:

Name	Value	Meaning
GET	0	Get value; a .GFDAT, .GFINF, or .GFSTA operation
BIC	1	A bit clear (BIC) operation
BIS	2	A bit set (BIS) operation
MOV	3	A word move (MOV) operation
	4-177	Reserved for Digital
USER	ucode	Reserved for the user

ucode is the user operation code containing the value from the *type* parameter. Specify *ucode* only when the *type* parameter argument is USER.



Errors:

Special directory handlers define their own codes. For RT–11 file structure devices:

Code	Explanation
0	Channel is not available
1	File not found, or not a file-structured device. (If it is necessary to determine what condition returned the error code, issue a .DSTAT request to determine if a device is file structured)
2	Operation is invalid
3	Invalid EMT argument block
4	Invalid EMT argument block

SOB

Macro Expansion

The SOB macro simulates the SOB instruction (subtract one and branch if not equal) by generating the code:

```
DEC     REG
BNE     ADDR
```

You can use the SOB macro on all processors, but it is especially useful for processors that do not have the hardware SOB instruction. If you are running on a processor that supports the SOB instruction, simply eliminate the MACRO call to SOB (.MCALL SOB), and the SOB instruction executes. Note that SOB is not preceded by a dot (.).

SOB reg,addr

where:

reg	is the register whose contents will be decremented by 1
addr	is the location to branch to if the register contents do not equal 0 after the decrement

.SPCPS

.SPCPS Support

EMT 375, Code 41

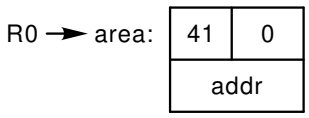
The .SPCPS (save/set mainline PC and PS) request allows a program's completion routine to change the flow of control of the mainline code.

.SPCPS area,addr

where:

area is the address of a two-word EMT argument block.

addr is the address of a three-word block in user memory.



Errors:

Code	Explanation
0	The program issued the .SPCPS call from the mainline code.
1	A previous .SPCPS request is outstanding.

When the program issues the .SPCPS request, the monitor saves the old mainline PS in the third word of the three-word block and the old mainline PC in the second word of the block. The monitor then changes the mainline PC to the contents of the first word of the block.

.SPFUN

EMT 375, Code 32

This request is used with certain device handlers to do device specific functions, such as rewind and backspace. It can be used with some disks to allow reading and writing of absolute sectors. It is commonly used to determine the size of devices of variable size.

.SPFUN area,chan,func,buf,wcnt,blk
[,crtn][,BMODE=strg][,CMODE=strg]

area is the address of a six-word EMT argument block

chan is a channel number.

func is the numerical code of the function to be performed; these codes must be negative

buf is the buffer address; this parameter must be set to zero if no buffer is required

wcnt is defined in terms of the device handler associated with the specified channel and in terms of the specified special function code

blk is also defined in terms of the device handler associated with the specified channel and in terms of the specified special function code

crtn is the entry point of a completion routine. If left blank, 0 is automatically inserted. This value is the same as for .READ, .READC, and .READW.

0 = wait I/O (.READW)
1 = real time (.READ)
Value >500 = completion routine

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space

Specifying *BMODE*:

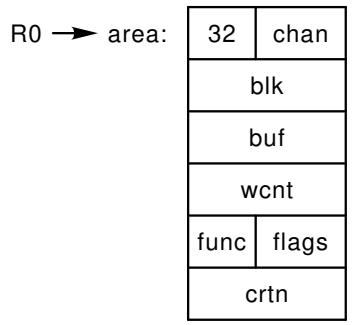
- Loads a word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.

CMODE=strg where *strg* is:

Value	Description
U	User space (default)
S	Supervisor space

Specifying *CMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *CMODE*.
- Specifies the *space* for the *crtn* argument.
- Is a valid option only for fully mapped monitors.



Errors:

Code	Explanation
0	Attempt to read or write past end-of-file, or invalid function value.
1	Hard error occurred on channel.
2	Channel is not open.

.SPND

EMT 374, Code 1

.SPND requests control execution of a job's mainline code; .SPND suspends the mainline and allows only completion routines (for I/O and mark time requests) to run.

.SPND

(.SPND) R0 =

1	0

Errors:
None.

.SRESET

EMT 352

.SRESET

The .SRESET (software reset) request:

- Cancels any messages sent by the job.
- Waits for all job I/O to complete, which includes waiting for all completion routines to run.
- Removes from memory any device handlers brought into memory via .FETCH calls by this job. Handlers loaded by the keyboard monitor LOAD command remain resident, as does the system device handler.
- Purges any currently open files. Files opened for output with .ENTER are never made permanent.
- Reverts to using only 16(decimal) I/O channels. Any channels defined with .CDFN are discarded. A .CDFN must be reissued to open more than 16 channels after a .SRESET is performed.
- Clears the job's .SPND/.RSUM counter.
- Resets the I/O queue to one element. A .QSET request must be reissued to allocate extra queue elements.
- Cancels all outstanding .MRKT requests.

Errors:
None.

.SYNCH

Macro Expansion

This macro call enables your program to issue programmed requests from an interrupt service routine. Code following the .SYNCH call runs at priority level 0 as a completion routine in the issuing job's context.

.SYNCH area[,pic]

where:

area is the address of a seven-word block that you must set aside for use by .SYNCH. This argument, area, represents a special seven-word block used by .SYNCH as a queue element. This is not the same as the regular area argument used by many other programmed requests. The user must not confuse the two; he should set up a unique seven-word block specifically for the .SYNCH request. The seven-word block appears as:

Word Offset	Contents
0 QS.LNK	RT–11 maintains this word; its contents should not be altered by the user
2 QS.JOB	The current job's number. Must be set up by the user program. Obtained by a .GTJB call or from the I/O queue element in a device handler
4	Unused.
6	Unused.
10 QS.ID	R0 contains this argument after successful return.
12 QS.SYN	Must be -1.
14 QS.CUP	Must be 0.

pic is an optional argument that, if non-blank, causes the .SYNCH macro to produce position-independent code for use by device drivers

.SYNCH

Errors:

The monitor returns to the location immediately following the .SYNCH if the .SYNCH was rejected. After failure of the .SYNCH routine, the routine is still unable to issue programmed requests, and R4 and R5 are available for use. An error is returned if another .SYNCH that specified the same seven-word block is still pending.

.TIMIO

Timeout

Macro Expansion

The .TIMIO macro issues the device time-out call in the handler I/O initiation section. This request schedules a completion routine to run after the specified time interval has elapsed. The completion routine runs in the context of the job indicated in the timer block.

.TIMIO tbk,hi,lo

where:

tbk is the address of the timer block, a seven-word pseudo timer queue element. (See timer block format shown under the .CTIMIO request.) You must set up the address of the completion routine in the seventh word of the timer block in a position-independent manner.

hi is the high-order word of a two-word time interval

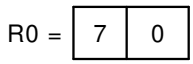
lo is the low-order word of a two-word time interval

.TLOCK

EMT 374, Code 7

The .TLOCK (test lock) request is used in a multijob environment to attempt to gain ownership of the USR. It is similar to .LOCK in that, if successful, the user job returns with the USR in memory (it is identical to .LOCK in the single job monitor). However, if a job attempts to .LOCK the USR while another job is using it, the requesting job is suspended until the USR is free. With .TLOCK, if the USR is not available, control returns immediately with the C bit set to indicate the .LOCK request failed.

.TLOCK



Errors:

Code	Explanation
0	USR is already in use by another job.

.TRPSET

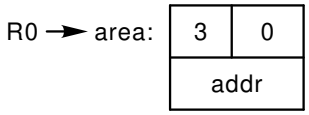
EMT 375, Code 3

.TRPSET enables a user job to intercept traps to 4 and 10 instead of having the job aborted with a *?MON-F-Trap to 4* or *?MON-F-Trap to 10* message. If .TRPSET is in effect when an error trap occurs, the user-specified routine is entered. The status of the carry bit on entry to the routine determines which trap occurred: carry bit clear indicates a trap to 4; carry bit set indicates a trap to 10. The user routine should exit with an RTI instruction. Traps are not intercepted by the .TRPSET request, but they do cause job abort and a printout of the message *?MON-F-Trap to 4*.

.TRPSET area,addr

where:

area	is the address of a two-word EMT argument block
addr	is the address of the user's trap routine. If an address of 0 is specified, trap interception is disabled



Errors:
None.

.TTINR/.TTYIN

EMT 340

The requests .TTINR and .TTYIN transfer a character from the console terminal to the user program. The character thus obtained appears right-justified (even byte) in R0. The user can cause the characters to be returned in R0 only, or in R0 and other locations.

.TTINR

.TTYIN char

where:

char is the location where the character in R0 is to be stored. If *char* is specified, the character is in both R0 and the address represented by *char*.

The expansion of .TTINR is:

```
EMT     340
```

The expansion of .TTYIN is:

```
EMT     340
BCS     .-2
```

If no characters or lines are available when an EMT 340 is executed, return is made with the carry bit set. The implication of these calls is that .TTYIN causes a tight loop waiting for a character/line to appear, while the user can either wait or continue processing using .TTINR.

.TTINR does not return the carry bit set, unless TCBIT$ of the job status word (JSW) was on when the request was issued.

The choice of two modes of doing console terminal input is determined by TTSPC$ of the job status word. If TTSPC$ is 0, normal I/O is performed. In this mode, the following conditions apply:

- The monitor echoes all characters typed.
- CTRL/U and the DELETE key perform line deletion and character deletion, respectively.
- A carriage return, line feed, CTRL/Z, or CTRL/C must be typed before characters on the current line are available to the program. As you type these, characters on the line are sequentially passed to the user program.

If TTSPC$ is 1, the console is in special mode. The effects are:

- The monitor does not echo characters typed except for CTRL/C and CTRL/O.
- CTRL/U and the DELETE key do not perform special functions.
- Characters are immediately available to the program.

In special mode, the user program must echo the characters received. However, CTRL/C and CTRL/O are acted on by the monitor in the usual way. TTSPC$ in the JSW must be set by the user program. This bit is cleared when the program terminates.

Regardless of the setting of TTSPC$, when a carriage return is entered, both carriage return and line feed characters are passed to the program; if TTSPC$ is 0, these characters will be echoed.

Errors:

Code	Explanation
0	No characters available in ring buffer.

.TTYOUT/.TTOUTR

EMT 341

The requests .TTYOUT and .TTOUTR cause a character to be transmitted to the console terminal. The difference between the two requests, as in the .TTYIN/.TTINR requests, is that if there is no room for the character in the monitor's buffer, the .TTYOUT request waits for room before proceeding, while the .TTOUTR does not wait for room and the character is not output.

.TTYOUT char

.TTOUTR

where:

char is the location containing the character to be loaded in R0 and printed. If not specified, the character in R0 is printed. Upon return from the request, R0 still contains the character

The expansion of .TTYOUT is:

```
EMT     341
BCS     .-2
```

The expansion of .TTOUTR is:

```
EMT     341
```

If the carry bit is set when execution of the .TTOUTR request is completed, it indicates that there is no room in the buffer and that no character was output. .TTOUTR normally does not return the carry bit set. Instead, the job is blocked until room is available in the output buffer. If a job requires execution to continue and the carry bit to be returned, it must turn on TCBIT$ of the Job Status Word before issuing the request.

The .TTINR and .TTOUTR requests have been supplied to help those users who want to continue rather than suspend program execution until a console operation is complete. With these modes of I/O, if a no-character or no-room condition occurs, the user program can continue processing and try the operation again at a later time.

If a foreground job leaves TCBIT$ set in the Job Status Word, any further foreground .TTYIN or .TTYOUT requests cause the system to lock out the background until a character is available. Note also that each job in the foreground/background environment has its own Job Status Word, and therefore can be in different terminal modes independently of the other job.

Errors:

Code	Explanation
0	Output ring buffer full.

.TWAIT

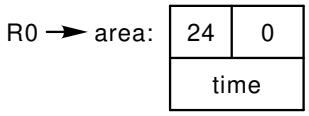
EMT 375, Code 24

The .TWAIT request suspends the user's job for a specified length of time. .TWAIT requires a queue element and should be a consideration when the .QSET request is issued.

.TWAIT area,time

where:

area	is the address of a two-word EMT argument block
time	is a pointer to two words of time (high order first, low order second), expressed in ticks



Errors:

Code	Explanation
0	No queue element was available.

.UNLOCK

EMT 347

The .UNLOCK request releases the User Service Routine (USR) from memory if it was placed there with a .LOCK request.

.UNLOCK

Errors:
None.

.UNMAP

Mapping

EMT 375, Code 36, Subcode 5

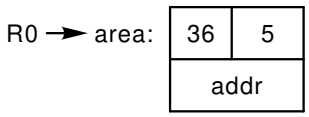
The .UNMAP request unmaps a window and makes inaccessible that portion of the program's virtual address space. When an unmap operation is performed for a virtual job, attempts to access the unmapped address space cause a memory management fault. For a privileged job, the default (Kernel) mapping is restored when a window is unmapped.

.UNMAP area,addr

where:

area is the address of a two-word argument block

addr is the address of the window control block that describes the window to be unmapped



Errors:

Code	Explanation
3	An illegal window identifier was specified.
5	The specified window was not already mapped.

.UNPROTECT

EMT 375, Code 31, Subcode 1

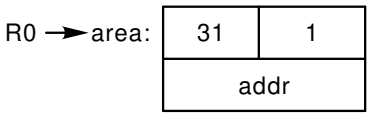
The .UNPROTECT request is the complement of the .PROTECT request. It cancels any protected vectors in the 0 to 476 area. An attempt to unprotect a vector that a job has not protected is ignored.

.UNPROTECT area,addr

where:

area is the address of a two-word EMT argument block

addr is the address of the protected vector pair that is going to be canceled. The argument *addr* must be a multiple of four, and must be less than or equal to 474_8



Errors:

Code	Explanation
1	Address (addr) is greater than 474_8 or is not a multiple of four.

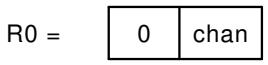
.WAIT

EMT 374, Code 0

The .WAIT request suspends program execution until all input/output requests on the specified channel are completed. The .WAIT request, combined with the .READ/.WRITE requests, makes double buffering a simple process.

.WAIT also conveys information through its error returns. An error is returned if the channel is not open or the last I/O operation resulted in a hardware error.

.WAIT chan



Errors:

Code	Explanation
0	Channel specified is not open.
1	Hardware error occurred on the previous I/O operation on this channel.

.WDBBK

Mapping

Macro Expansion

The .WDBBK macro defines symbols for the window definition block and reserves space for it. Information provided to the arguments of this macro permits the creation and mapping of a window through the use of the .CRAW request. Note that .WDBBK automatically invokes .WDBDF.

.WDBBK wnapr,wnsiz[,wnrid][,wnoff][,wnlen][,wnsts]

where:

wnapr is the number of the Active Page Register set that includes the window's base address. A window must start on a 4K-word boundary. The valid range of values is from 0 through 7

wnsiz is the size of this window (expressed in 32-word units)

wnrid is the identification for the region to which this window maps. Use the value of R.GID from the region definition block for this argument after you create the region to which this window must map

wnoff is the offset into the region at which to start mapping this window (expressed in 32-word units). The default is 0, which means that the window starts mapping at the region's base address

wnlen is the amount of this window to map (expressed in 32-word units). The default value is 0, which maps as much of the window as possible

wnsts is the window status word. Set bit 8, called WS.MAP, to cause .CRAW to perform an implied mapping operation

.WDBDF

Mapping

Macro Expansion

The .WDBDF macro defines the symbolic offset names for the window definition block and the names for the window status word bit patterns. In addition, this macro also defines the length, but doesn't reserve any space for the definition block. (See .WDBBK).

.WDBDF

where:

The .WDBDF macro expands as follows:

```
W.NID   =: 0            ;Window ID
W.NAPR  =: 1            ;PAR number
W.NBAS  =: 2.           ;Base address
W.NSIZ  =: 4.           ;Window size
W.NRID  =: 6.           ;Region ID
W.NOFF  =: ^o10         ;Window offset
W.NLEN  =: ^o12         ;Window length
W.NSTS  =: ^o14         ;Window status
W.NLGH  =: ^o16         ;Length of WDB
WS.CRW  =: ^o100000     ;Window created
WS.UNM  =: ^o40000      ;One or more windows unmapped
WS.ELW  =: ^o20000      ;One or more windows eliminated
WS.DSI  =: ^o10000      ;D-space inactive WS.D=WS.I=1
WS.IDD  =: ^o4000       ;D-space window different WS.D=WS.I=1
WS.OVR  =: ^o2000       ;reserved
WS.RO   =: ^o1000       ;Window is read only
WS.MAP  =: ^o400        ;Create and map window
WS.SPA  =: ^o14         ;Bit field for address space(s)
WS.D    =: ^o10         ;Map into data space
WS.I    =: ^o4          ;Map into instruction space
WS.MOD  =: ^o3          ;Field to indicate mode
WS.U    =: ^o0          ;User
WS.S    =: ^o1          ;Supervisor
WS.C    =: ^o2          ;Current
```

.WRITE

EMT 375, Code 11

The .WRITE request transfers a specified number of words from memory to the specified channel. Control returns to your program immediately after the request is queued.

.WRITE area,chan,buf,wcnt,blk[,BMODE=strg]

where:

area is the address of a five-word EMT argument block

chan is a channel number.

buf is the address of the memory buffer to be used for output

wcnt is the number of words to be written

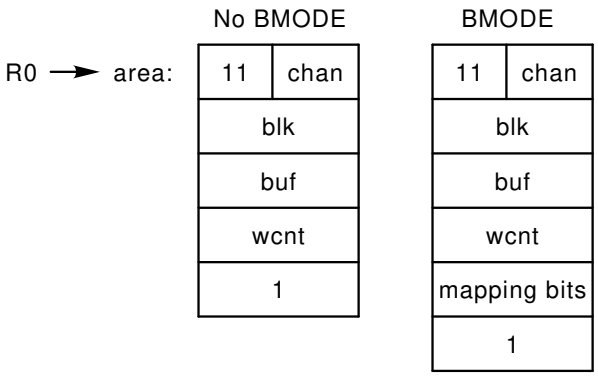
blk is the block number to be written. For a file-structured .LOOKUP or .ENTER, the block number is relative to the start of the file. For a non-file-structured .LOOKUP or .ENTER, the block number is the absolute block number on the device. The user program should normally update *blk* before it is used again. Some devices, such as LP, may assign the *blk* argument special meaning. For example, if *blk* = 0, LP: issues a form feed

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	Attempted to write past end-of-file.
1	Hardware error.
2	Channel was not opened.

.WRITC

EMT 375, Code 11

The .WRITC request transfers a specified number of words from memory to a specified channel. Control returns to the user program immediately after the request is queued. Execution of the user program continues until the request is fulfilled, then control passes to the routine specified in the request. When the completion routine encounters a RETURN, control returns to the user program.

.WRITC area,chan,buf,wcnt,crtn,blk[,BMODE][,CMODE]

where:

area	is the address of a five-word EMT argument block
chan	is a channel number.
buf	is the address of the memory buffer to be used for output
wcnt	is the number of words to be written
crtn	is the address (>500 octal) of the completion routine to be entered
blk	is the block number to be written. For a file-structured .LOOKUP or .ENTER, the block number is relative to the start of the file. For a non-file-structured .LOOKUP or .ENTER, the block number is the absolute block number on the device. Your program should normally update *blk* before it is used again. See the *RT–11 Device Handlers Manual* for the significance of the block number for devices such as line printers.

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

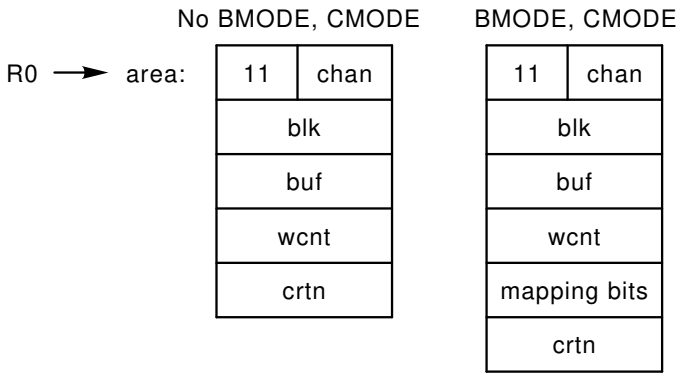
- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.

CMODE=strg where *strg* is:

Value	Description
U	User space (default)
S	Supervisor space

Specifying *CMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *CMODE*.
- Specifies the *space* for the *crtn* argument.
- Is a valid option only for fully mapped monitors.



Errors:

Code	Explanation
0	End-of-file on output. Tried to write outside limits of file.
1	Hardware error occurred.
2	Specified channel is not open.

.WRITW

EMT 375, Code 11

The .WRITW request transfers a specified number of words from memory to the specified channel. Control returns to your program when the .WRITW is complete.

.WRITW area,chan,buf,wcnt,blk[,BMODE=strg]

where:

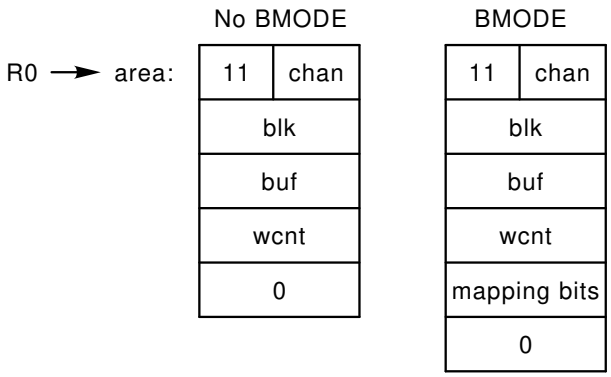
area	is the address of a five-word EMT argument block
chan	is a channel number in the range 0 to 376_8
buf	is the address of the buffer to be used for output
wcnt	is the number of words to be written. The number must be positive
blk	is the block number to be written. For a file-structured .LOOKUP or .ENTER, the block number is relative to the start of the file. For a non-file-structured .LOOKUP or .ENTER, the block number is the absolute block number on the device. Your program should normally update *blk* before it is used again. See the *RT–11 Device Handlers Manual* for the significance of the block number for devices such as line printers and paper tape punchers

BMODE=strg where *strg* is:

Value	Description
UD	User data space (default)
UI	User instruction space
SD	Supervisor data space
SI	Supervisor instruction space
CD	Kernel data space
CI	Kernel instruction space

Specifying *BMODE*:

- Loads an additional word in the EMT request block area containing a bit pattern matching the code specified for *BMODE*.
- Specifies *mode* and *space* for the *buff* argument.
- Is a valid option only in a fully mapped environment.



Errors:

Code	Explanation
0	Attempted to write past EOF.
1	Hardware error.
2	Channel was not opened.

Chapter 6

Subroutines

This chapter presents all SYSLIB functions and subroutines in alphabetical order by generic name. Because some subroutines can be called either as subroutines or functions, they are presented here under the generic form; for example, *READ. This convention simplifies lookup. An I-prefixed name indicates use as a function; an M-prefixed name indicates mapping is specified.

Each description briefly defines the subroutine or function; gives its argument list, and defines each parameter and argument contained in the argument list. Function results and errors are listed for each, as appropriate. For more detailed information and specific examples, see the *RT–11 System Subroutine Library Manual*.

Subroutine/Function Syntax

CALL ABTIO (chan)
i = IABTIO (chan)
Aborts I/O on a specified channel.

a = AJFLT (jsrc)
i = IAJFLT (jsrc,ares)
Converts the specified INTEGER*4 value to REAL*4 and returns specified FORTRAN logical unit.

CALL CALL$F
Saves contents of general registers R1–R4 across a call to another routine that might destroy the contents of those registers.

CALL CHAIN (dblk,var,wcnt)
Transfers control and passes specified information to another background program.

CALL CHCPY (chan,ochan[,jobblk])
i = ICHCPY (chan,ochan[,jobblk])
Opens a channel for input, logically connecting it to a file that is currently open by another job.

CALL CLOSEC (chan[,i])
i = ICLOSE (chan)
Closes the specified channel.

CALL CLOSZ (chan, size)
i = ICLOSZ (chan,size)
Terminates activity on specified channel and frees it for use in another operation.

CALL CMAP (ival [,iold][,ierr])
i = ICMAP (ival [,iold])
Controls mapping for Supervisor mode and I-D space.

CALL CMKT(id,time)
i = ICMKT (id,time)
Cancels an unexpired mark time request.

CALL CONCAT (a,b,out[,len[,err]])
Concatenates two variable-length strings.

CALL CNTXS (addr)
i = ICNTXS (addr)
Saves a list of address locations when a transition is made from one running job to another.

CALL CRAW (iwdb [,ierr])
i = ICRAW (iwdb)
Mapping routine for Supervisor mode, I-D space; creates an address into existing memory region.

CALL CRRG (idrd [,ierr])
i = ICRRG (irdb)
Mapping routine for Supervisor mode, I-D space; allocates or attaches to a region in physical memory for use by requesting job.

CALL CSI (filspc,deftyp[,cstring],[,option],n)
i = ICSI (filspc,deftyp[,cstring],[,option],n)
Calls the RT–11 CSI in special mode to decode file specifications and options.

CALL CSTAT (chan,addr)
i = ICSTAT (chan,addr)
Returns the status of a specified channel.

CALL CVTTIM (time,hrs,min,sec,tick)
Converts a 2-word internal format time to hours, minutes, seconds, and ticks.

CALL DATE (array[(i)][,opdate])
CALL DATE4Y (array[(i)][,opdate])
The DATE and DATE4Y subroutines display date or format a date you specify.

CALL DELET (chan,dblk[,seqnum])
i = IDELET (chan,dblk[,seqnum])
Deletes the file from the specified device.

CALL DEVICE (illst[,link])
i = IDEVIC (illst[,link])
Sets up a list of addresses to be loaded with specific values when the program is terminated.

CALL DJFLT (jsrc,dres)
i = IDJFLT (jsrc,dres)
Converts an INTEGER*4 value into a REAL*8 (DOUBLE PRECISION) value and returns that result as the function value.

CALL DSTAT (devnam,cblk)
i = IDSTAT (devnam,cblk)
Returns the status of the specified device.

CALL ELAW (iwdb [,ierr])
i = IELAW (iwdb [,ierr])
Eliminates a virtual address window.

CALL ELRG (irdb [,ierr])
i = ELRG (irdb [,ierr])
Eliminates a dynamic region of physical memory and returns the memory to a free list.

CALL ENTER (chan,dblk,length[,seqnum])
i = IENTER (chan,dblk,length[,seqnum])
Creates a new file for output.

CALL FPROT (chan,filespec,prot)
i = IFPROT (chan,filespec,prot)
Sets or removes file protection.

CALL FREER (work[,,<type>])
i = IFREER (work[,,<type>])
Detaches from or eliminates a global region attached to or created by IGETR.

CALL GCMAP (iold [,ierr])
i = IGCMAP (iold [,ierr])
Returns the previous CMAP status.

CALL GETR (arguments)
i = IGETR (arguments)
(work,char,name,addr[,csize][,offset[,msize]] [,chan[,blk]][,file[,blk]][,sbrtn,-1])
Attaches to a specified global region.

CALL GFDAT (chan,dblk,idate)
i = IGFDAT (chan,dblk, idate)
Returns file creation date from file's directory entry (E.DATE) word.

CALL GFINF (chan,dblk,offset,ival)
i = IGFINF (chan,dblk,offset,ival)
Returns word contents of the directory entry offset specified from a file's directory entry.

CALL GFSTA (chan,dblk,istat)
i = IGFSTA (chan,dblk,istat)
Returns word contents of the directory entry status word (E.STAT) from a file's directory entry.

GICLOS (status,lun)
Ends GIDIS connection to PI handler.

GIOPEN (status,lun,message,msglen,devtype,driver)
Initiates contact with PI handler for GIDIS.

GIREAD (status,lun,buffer,buflen)
Returns a report from GIDIS requested by a GIWRIT.

GIWRIT (status,lun,message,msglen)
Sends a buffer of GIDIS command data to PI handler.

CALL GMCX (iwdb [,ierr])
i = IGMCX (iwdb [,ierr])
Returns mapping status of an extended memory window.

CALL GTDIR (arguments)
i = IGTDIR (arguments)
(wksize,wkarea,chan,buffer[,header][,dblk][,string][,stvalu][,stmask] [,datrel][,datewd][,resrv1]
[,resrv2][,stofst])
Sets up parameters for wildcard directory search on RT–11 file-structured volume or logical disk file.

CALL GTDUS (dev,ichan,ibuf[,iunit][,itype][,iwork][,isize])
i = IGTDUS (dev,ichan,ibuf[,iunit][,itype][,iwork][,isize])
Provides information about a specified MSCP (DU) or TMSCP (MU) class device unit.

CALL GTIM (itime)
Returns current time of day.

CALL GTJB (addr[,jobblk][,i]])
i = IGTJB (addr[,jobblk][,i]])
Returns information about a job in the system.

CALL GTLIN (arguments)
i = IGTLIN (arguments)
(result[,prompt][,term][,plain])
Transfers a line of input to your program from the terminal or from an active indirect file.

CALL HERR ( )
i = IHERR ( )
Turns off error interception (ISERR); lets the monitor abort a job and generate an error message under fatal error conditions.

i = IADDR (arg)
Returns 16-bit absolute memory address of its arguments as the integer function value.

i = IDATE (mon,day,year[,opdate])
Returns three INTEGER*2 values for system month, day, and year.

i = IDCOMP (date1[,date2])
Compares two RT–11 date words and returns an integer value indicating relationship of the dates.

i = IFWILD (tststr,matstr[,explct])
Tests file specification string for match against a maximum of eight wildcard file specifications.

i = IGTENT (wkarea,entry[,entofs][,filblk][,ascnam])
Returns next directory entry matching criteria specified in the IGTDIR function.

i = IJCVT (jsrc[,ires])
Converts an INTEGER*4 value to INTEGER*2 format.

CALL INDEX (a,pattrn[,i],m)
i = INDEX (a,pattrn[,i],m)
Searches source string for occurrence of pattern string and returns character position of first occurrence within source.

CALL INSERT (in,out,i[,m])
Replaces a portion of one string with another string.

i = IPEEK (iaddr)
Returns the contents of the word located at a specified 16-bit address in the current job's address space.

i = IPEEKB (iaddr)
Returns the contents of the byte located at a specified 16-bit address in the current job's address space.

CALL IRAD50 (icnt,input,output)
n = IRAD50 (icnt,input,output)
Converts ASCII characters to Radix–50 and returns the number of characters converted.

i = ISPY (ioff)
Returns the integer value of the word at a specified offset from the RT–11 resident monitor.

i = ISWILD (tststr,matstr[,term][,case][,explct])
Checks two ASCII strings for matching, using wildcards or not.

i = ITLOCK ( )
Indicates whether the USR is currently in use by another job and performs a lock if possible.

i = ITTINR ( )
Transfers a character from the terminal to the user program.

i = ITTOUR (char)
Transfers a character from the user program to the terminal.

i = IWEEKD (month,iday,iyear)
Returns an integer value representing a specified year, month and day.

CALL JADD (jopr1,jopr2,ires)
i = JADD (jopr1,jopr2,ires)
Computes the sum of two INTEGER*4 values.

CALL JAFIX (asrc,jres)
i = JAFIX (asrc,jres)
Converts a REAL*4 value to INTEGER*4.

i = JCMP (jopr1,jopr2)
Compares two INTEGER*4 values and returns an INTEGER*2 value that reflects the signed comparison result.

i = JDFIX (dsrc,jres)
Converts a REAL*8 value to INTEGER*4.

CALL JDIV (jopr1,jopr2,jres[,jrem])
i = JDIV (jopr1,jopr2,jres[,jrem])
Computes the quotient of two INTEGER*4 values.

CALL JICVT (isrc,jres)
i = JICVT (isrc,jres)
Converts an INTEGER*2 value to INTEGER*4.

CALL JJCVT (jsrc)
Converts 2-word internal time formats to INTEGER*4 format, and vice versa.

CALL JMOV (jsrc,jdest)
i = JMOV (jsrc,jdest)
Assigns an INTEGER*4 value to a variable.

CALL JMUL (jopr1,jopr2,jres)
i = JMUL (jopr1,jopr2,jres)
Computes the product of two INTEGER*4 values.

CALL JREAD (wcnt,buff,jblock,chan[,area][,BMODE=strg])
i = JREAD (wcnt,buff,jblock,chan[,area][,BMODE=strg])
Transfers into memory words from MSCP device associated with channel. BMODE specifies mapping.

CALL JREADC (wcnt,buff,jblock,chan,[,area],crtn[,BMODE=strg][,CMODE=strg])
i = JREADC (wcnt,buff,jblock,chan,[,area],crtn[,BMODE=strg][,CMODE=strg])
Transfers into memory words from MSCP device associated with channel. BMODE and CMODE specify mapping, Supervisor address.

CALL READF (wcnt,buff,jblock,chan,[,area],lblk,frtn)
i = JREADF (wcnt,buff,jblock,chan,[,area],lblk,frtn)
Transfers into memory words from an MSCP device, then enters *frtn* completion routine.

CALL JREADW(wcnt,buff,jblock,chan[,area][,BMODE=strg])
i = JREADW(wcnt,buff,jblock,chan[,area][,BMODE=strg])
Transfers into memory words from an MSCP device; BMODE specifies mapping.

CALL JSUB (jopr1,jopr2,jres)
i = JSUB (jopr1,jopr2,jres)
Computes the difference between two INTEGER*4 values.

CALL JTIME (hrs,min,sec,tick,time)
Converts time specified into internal two-word format time.

CALL JWRITE (wcnt,buff,jblock,chan[,area][,BMODE=strg])
i = JWRITE (wcnt,buff,jblock,chan[,area][,BMODE=strg])
Transfers into memory words from MSCP device associated with channel. BMODE specifies mapping.

CALL JWRITC (wcnt,buff,jblock,chan[,area],crtn[,BMODE=strg][,CMODE=strg])
i = JWRITC (wcnt,buff,jblock,chan[,area],crtn[,BMODE=strg][,CMODE=strg])
Transfers into memory words from MSCP device associated with channel. BMODE and CMODE specify mapping, Supervisor address.

CALL JWRITF (wcnt,buff,jblock,chan[,area],lblk,frtn)
i = JWRITF (wcnt,buff,jblock,chan[,area],lblk,frtn)
Transfers into memory words from an MSCP device; then enters *frtn* completion routine.

CALL JWRITW(wcnt,buff,jblock,chan[,area][,BMODE=strg)
i = JWRITW(wcnt,buff,jblock,chan[,area][,BMODE=strg)
Transfers into memory words from an MSCP device. BMODE specifies mapping.

i = KPEEK (iaddr[,ierr])
Returns contents of word located at specified 16-bit address.

CALL KPOKE (iaddr,ivalue[,type][,ierr])
i = KPOKE (iaddr,ivalue[,type][,ierr])
Stores specified 16-bit integer value at Kernel-mapped address.

i = LEN (a)
Returns the number of characters in string contained in an array *a*.

CALL LOCK
Keeps USR permanently resident until an UNLOCK is executed.

CALL LOOKUP (chan,jobdes)
i = LOOKUP (chan,jobdes)
Opens an existing system job for I/O with the specified channel.

LOOKUP (chan,dblk[,seqnum])
i = LOOKUP (chan,dblk[,seqnum])
Opens an existing file for I/O with the specified channel.

CALL MAP (iwdb [,ierr])
i = MAP (iwdb)
Maps a previously defined address window.

CALL MRKT (id,crtn,time[,CMODE=strg])
i = MRKT (id,crtn,time[,CMODE=strg])
Schedules an assembly language routine to be entered after a specified interval. CMODE specifies Supervisor address.

CALL MSDS (ival [,iold])
i = MSDS (ival [,iold][,ierr])
Controls the linkage of Supervisor and User data space.

CALL MTATCH (unit[,addr,][,jobnum][,AMODE=strg)
i = MTATCH (unit[,addr,][,jobnum][,AMODE=strg)
Attaches a specific terminal. AMODE specifies S-D address.

CALL MTDTCH (unit)
i = MTDTCH (unit)
Detaches and makes available a specific terminal.

CALL MTGET (unit,addr[,jobnum])
i = MTGET (unit,addr[,jobnum])
Gets status information about a specific terminal in multiterminal system.

CALL MTIN (unit,char[,chrcnt,ocnt])
i = MTIN (unit,char[,chrcnt,ocnt])
Transfers a character from a specific terminal to the user program.

CALL MTOUT (unit,char[,chrcnt,ocnt])
i = MTOUT (unit,char[,chrcnt,ocnt])
Transfers a character to a specific terminal.

CALL MTPRNT (unit,string)
i = MTPRNT (unit,string)
Outputs an ASCII string to a specific terminal.

CALL MTRCTO (unit)
i = MTRCTO (unit)
Resets CTRL/O for a specific terminal.

CALL MTSET (unit,addr)
i = MTSET (unit,addr)
Sets terminal and line characteristics.

CALL MTSTAT (addr)
i = MTSTAT (addr)
Returns multiterminal system status in an 8-word status block.

CALL MWAIT
Suspends main program execution until messages between jobs have been processed.

CALL POKE (iaddr,ivalue)
i = IPOKE (iaddr,ivalue)
Stores specified 16-bit integer in a 16-bit address space.

CALL POKEB (iaddr,ivalue)
i = IPOKEB (iaddr,ivalue)
Stores specified 8-bit integer in a 16-bit address space.

CALL PRINT (string)
Prints an ASCII string at the terminal.

CALL PROTE (addr)
i = IPROTE (addr)
Obtains exclusive control of a two-word vector.

CALL PURGE (chan)
Deactivates a channel.

CALL PUT (ioff,value[,operation][,error])
i = IPUT (ioff,value[,operation][,error])
Replaces value of a monitor fixed offset.

CALL R50ASC (icnt,input,output)
Converts Radix–50 characters to ASCII.

a = RAD50 (input)
Converts six ASCII characters and returns a REAL*4 result that is the two-word Radix–50 value.

CALL RANDU (iseed1,iseed2)
Pseudo-random number generator.

f = RAN (iseed)
f = RAN (iseed1 or iseed2)
Pseudo-random number generator.

CALL RCHAIN (flag,var,wcnt)
Lets program determine if it has been chained-to and to access variables passed across a chain.

CALL RCTRLO
Resets effect of any CTRL/O command.

CALL = RCVD (buff,wcnt)
i = IRCVD (buff,wcnt)
CALL = MRCVD (buff,wcnt[,BMODE=strg])
i = MRCVD (buff,wcnt[,BMODE=strg])
Receives data and returns to user program immediately.
MRCVD specifies mapping.

CALL RCVDC (buff,wcnt,crtn)
i = IRCVDC (buff,wcnt,crtn)
CALL MRCVDC (buff,wcnt,crtn[,BMODE=strg][,CMODE=strg])
i = MRCVDC (buff,wcnt,crtn[,BMODE=strg][,CMODE=strg])
Receives data and enters an assembly language completion routine.
MRCVDC specifies mapping.

CALL IRCVDF (buff,wcnt,area,frtn)
i = IRCVDF (buff,wcnt,area,frtn)
Receives data and enters a FORTRAN subprogram.
MRCVDF specifies mapping.

CALL RCVDW (buff,wcnt)
i = IRCVDW (buff,wcnt)
CALL MRCVDW (buff,wcnt[,BMODE=strg])
i = MRCVDW (buff,wcnt[,BMODE=strg])
Receives data, waits for completion, and returns to the user program.
MRCVDW specifies mapping.

CALL READ (wcnt,buff,blk,chan)
i = IREAD (wcnt,buff,blk,chan)
CALL MREAD (wcnt,buff,blk,chan[,BMODE=strg])
i = MREAD (wcnt,buff,blk,chan[,BMODE=strg])
Transfers data on the specified channel to a memory buffer and returns control to the user program; MREAD specifies mapping.

CALL READC (wcnt,buff,blk,chan,crtn)
i = IREADC (wcnt,buff,blk,chan,crtn)
CALL MREADC (wcnt,buff,blk,chan,crtn[,BMODE=strg][,CMODE=strg])
i = MREADC (wcnt,buff,blk,chan,crtn[,BMODE=strg][,CMODE=strg])

Transfers data on the specified channel to a memory buffer and enters an assembly language completion routine; MREADC specifies mapping.

CALL READF (wcnt,buff,blk,chan,area,frtn)
i = IREADF (wcnt,buff,blk,chan,area,frtn)

Transfers data on the specified channel to memory buffer and enters a FORTRAN program.

CALL READW (wcnt,buff,blk,chan)
i = IREADW (wcnt,buff,blk,chan)
CALL MREADW (wcnt,buff,blk,chan[,BMODE=strg])
i = MREADW (wcnt,buff,blk,chan[,BMODE=strg])

Transfers data on the specified channel to a memory buffer and returns control to the user program only after the transfer is complete; MREADW specifies mapping.

CALL RENAM (chan,dblk)
i = IRENAM (chan,dblk)

Changes the name of a specified file.

CALL REOPN (chan,cblk)
i = IREOPN (chan,cblk)

Identifies a specified channel with a file on which an ISAVES was performed.

CALL REPEAT (in,out,i[,len[,err]])

Concatenates a specified string with itself to produce an indicated number of copies.

CALL RESUME

Lets a mainline job resume after being suspended with a SUSPND call.

CALL SAVES (chan,cblk)
i = ISAVES (chan,cblk)

Stores five words of channel status information into an array.

CALL SCCA [([iflag][,itype])]
i = ISCCA ([iflag][,itype])
CALL MSCCA [([iflag][,itype][,AMODE=strg])]
i = MSCCA ([iflag][,itype][,AMODE=strg])

Inhibits a CTRL/C abort; indicates that a CTRL/C is active; distinguishes between a single and double CTRL/C. MSCCA specifies mapping.

CALL SCHED (hrs,min,sec,tick,area,id,crtn)
i = ISCHED (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be entered at the specified time of day as an asynchronous completion routine.

CALL SCOMP (a,b,i)
i = SCOMP (a,b)
Compares two character strings and returns the integer result of the comparison.

CALL SCOPY (in,out,[,len][,err])
Copies a character string from one array to another.

CALL SDAT (buff,wcnt)
i = ISDAT (buff,wcnt)
CALL MSDAT (buff,wcnt[,BMODE=strg])
i = MSDAT (buff,wcnt[,BMODE=strg])
Transfers the specified number of words from one job to the other and returns control immediately to the user program. BMODE specifies mapping.

CALL SDATC (buff,wcnt,crtn)
i = ISDATC (buff,wcnt,crtn)
CALL MSDATC (buff,wcnt,crtn[,BMODE=strg][,CMODE=strg])
i = MSDATC (buff,wcnt,crtn[,BMODE=strg][,CMODE=strg])
Transfers the specified number of words from one job to the other and returns control immediately to the user program. MREADC specifies mapping.

CALL SDATF (buff,wcnt,frtn)
i = ISDATF (buff,wcnt,area,frtn)
Transfers the specified number of words from one job to the other and enters a FORTRAN program.

CALL SDATW (buff,wcnt)
i = ISDATW (buff,wcnt)
CALL MSDATW (buff,wcnt[,BMODE=strg])
i = MSDATW (buff,wcnt[,BMODE=strg])
Transfers the specified number of words from one job to the other and returns control to the user program only after the transfer is complete. MSDATW specifies mapping.

CALL SDTTM (date,hitime,lotime)
i = ISDTTM (date,hitime,lotime)
Sets system date and time.

CALL SERR ( )
i = ISERR ( )

Inhibits monitor from aborting job; causes error return to EMT that caused the error.

CALL SETCMD (string)

Lets a user program pass a command line to the keyboard monitor to be executed after the program exits.

CALL SFDAT (chan,dblk,idate)
i = ISFDAT (chan,dblk,idate)

Changes the creation date of an RT–11 file.

CALL SFINF (chan,dblk,value,oper,offset[,iold])
i = ISFINF (chan,dblk,value,oper,offset[,iold])

Saves and modifies the contents of specified directory entry offset.

CALL SFSTA (chan,dblk,value,oper[,iold])
i = ISFSTA (chan,dblk,value,oper[,iold])

Saves and modifies the contents of directory entry status word (E.STAT).

CALL SLEEP (hrs,min,sec,tick)
i = ISLEEP (hrs,min,sec,tick)

Suspends mainline execution of the running job for a specified amount of time.

CALL SPCPS (addr)
i = ISPCPS (addr)

Lets a program completion routine change control flow of mainline code.

CALL SPFN (code,chan[,wcnt,buff,blk])
i = ISPFN (code,chan[,wcnt,buff,blk])
CALL MSPFN (code,chan[,wcnt,buff,blk][,BMODE=strg])
i = MSPFN (code,chan[,wcnt,buff,blk][,BMODE=strg])

Queues the specified operation and returns control to the user program immediately. MSPFN specifies mapping.

CALL SPFNC (code,chan,wcnt,buff,blk,crtn)
i = ISPFNC (code,chan,wcnt,buff,blk,crtn)
CALL MSPFNC (code,chan,wcnt,buff,blk,crtn[,BMODE=strg][,CMODE=strg])
i = MSPFNC (code,chan,wcnt,buff,blk,crtn[,BMODE=strg][,CMODE=strg])

Queues the specified operation and enters an assembly language program completion routine. MSPFNC specifies mapping.

CALL SPFNF (code,chan,wcnt[,buff][,blk],area,crtn)
i = ISPFNF (code,chan,wcnt[,buff][,blk],area,crtn)
Queues the specified operation and enters a FORTRAN subprogram.

CALL SPFNW (code,chan[,wcnt,buff,blk])
i = ISPFNW (code,chan[,wcnt,buff,blk])
CALL MSPFNW (code,chan[,wcnt,buff,blk][,BMODE=strg])
i = MSPFNW (code,chan[,wcnt,buff,blk][,BMODE=strg])
Queues the specified operation and returns control to the user after operation completes. MSPFNW specifies mapping.

CALL STRPAD (a,len,[,err])
Pads character string with right-most blanks until string is specified length.

CALL SUBSTR (in,out,i[,len])
Copies a substring from a specified string.

CALL SUSPND
Suspends main program execution of current job; completion routines continue to run.

CALL TIMASC (itime,strng)
Converts 2-word internal format time into an ASCII string.

CALL TIME (strng)
Returns the current system time of day as an 8-character ASCII string.

CALL TIMER (hrs,min,sec,tick,area,id,crtn)
i = ITIMER (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be run as asynchronous completion routine after specified interval has elapsed.

CALL TRANSL (in,out,r[,p])
Replaces one string with another after performing character modification.

CALL TRIM (a)
Removes trailing blanks from a character string.

CALL TWAIT (itime)
i = ITWAIT (itime)
Suspends main program execution of current job for specified interval.

CALL UNLOCK ( )
Releases USR from memory, if it was placed there by LOCK routine.

CALL UNMAP (iwdb[,ierr])
i = IUNMAP (iwdb)
Eliminates the mapping window.

CALL UNPRO (addr)
i = IUNPRO (addr)
Cancels protection for specified 2-word vector in 0–474 area.

CALL UNTIL (hrs,min,sec,tick)
i = IUNTIL (hrs,min,sec,tick)
Suspends the main program execution of the job until specified time of day. Completion routines continue to run.

CALL VERIFY (a,b,i)
i = IVERIF (a,b)
Checks that a string is composed entirely of characters from a second string.

CALL WAIT (chan)
i = IWAIT (chan)
Suspends execution of main program until all input/output operations on channel are complete.

CALL WRITE (wcnt,buff,blk,chan)
i = IWRITE (wcnt,buff,blk,chan)
CALL MWRITE (wcnt,buff,blk,chan[,BMODE=strg])
i = MWRITE (wcnt,buff,blk,chan[,BMODE=strg])
Transfers data on specified channel to a device and enters an assembly language completion routine. MWRITE specifies mapping.

CALL WRITC (wcnt,buff,blk,chan,crtn)
i = IWRITC (wcnt,buff,blk,chan,crtn)
CALL MWRITC (wcnt,buff,blk,chan,crtn[,BMODE=strg][,CMODE=strg])
i = MWRITC (wcnt,buff,blk,chan,crtn[,BMODE=strg][,CMODE=strg])
Transfers data on specified channel to a device and returns control immediately to user program. MWRITC specifies mapping.

CALL WRITF (wcnt,buff,blk,chan,area,frtn)
i = IWRITF (wcnt,buff,blk,chan,area,frtn)
Transfers data on specified channel to a device and enters a FORTRAN subprogram.

CALL WRITW (wcnt,buff,blk,chan)
i = IWRITW (wcnt,buff,blk,chan)
CALL MWRITW (wcnt,buff,blk,chan[,BMODE=strg])

i = MWRITW (wcnt,buff,blk,chan[,BMODE=strg])

Transfers data on specified channel to a device and returns control to the user program after the transfer is complete. MWRITW specifies mapping.

Chapter 7

Monitor Data Structures

This chapter contains data tables for control blocks, disk and file formats, RMON fixed offsets and queue elements. For more detailed information, see the *RT–11 System Internals Manual*, *RT–11 Device Handlers Manual*, and the *RT–11 Volume and File Formats Manual*.

Control Blocks

Table 7–1: Channel Status Word (CSW), .CSWDF

Bit	Symbol	Meaning
0	HDERR$	Hard error bit. 0 = No error. 1 = Hard error.
1-5	INDX$M	Index into the $PNAME table and other device tables.
6	RENAM$	RENAME flag. 0 = No RENAME is in progress. 1 = A RENAME operation is in progress.
7	DWRIT$	0 = The file was opened with a .LOOKUP. The monitor does not modify the directory when the file is closed. 1 = The file was opened with a .ENTER. The monitor modifies the directory when the file is closed.
8-12	DBLK$M	The number of the directory segment containing this entry.
13	EOF$	End-of-file (EOF) bit. 0 = No end-of-file. 1 = End-of-file was found on this channel.
14	RONLY$	Write not allowed.
15	ACTIV$	0 = The channel is free. 1 = The channel is active.

Table 7–2: I/O Channel Description (.CHNDF)

Name	Offset	Contents
C.CSW	0	Channel Status Word
C.SBLK	2	Starting block number of this file (0 if non-file-structured)
C.LENG	4	Length of file (if opened by .LOOKUP) Size of empty area (if opened by .ENTER)
C.USED	6	Highest block written
C.DEVQ	10	I/O count (pending requests)
C.UNIT	11	Device unit number
C.SIZ	12	Symbol whose value is the size of block in bytes

Table 7–3: Handler Prefix Area Format

The handler prefix area is contained in block 0 of each handler.

Name	Offset	Contents
H.SIZ	52	Size of handler in bytes
H.DVSZ	54	Size of device in 256-word blocks
H.DSTS	56	Default device status value
H.GEN	60	Sysgen options
H.BPTR	62	Pointer to primary bootstrap
H.BLEN	64	Length of primary bootstrap in bytes
H.READ	66	Pointer to read routine for bootstrap

Table 7–4: Region Definition Block (.RDBDF)

Offset	Symbol	Modifier	Contents
0	R.GID	Monitor's .CRRG routine	A unique region identification; a pointer to the region control block.
2	R.GSIZ	.RDBBK macro or user program	The size of the region you need, in 32_{10}-word units.

Table 7–4 (Cont.): Region Definition Block (.RDBDF)

Offset	Symbol	Modifier	Contents
4	R.GSTS	Monitor's .CRRG routine	The region status word.
6	R.GNAM	.RDBBK macro	2-word global region name.
12	R.GBAS	.RDBBK macro	Global region base address.

Table 7–5: Region Status Word (.RDBDF)

Bit	Symbol	Pattern	Meaning When Set
15	RS.CRR	100000	The monitor created the region successfully. The .CRRG routine sets this bit; the .ELRG routine clears it.
14	RS.UNM	40000	One or more windows were unmapped as a result of eliminating the region. The .ELRG routine sets this bit when necessary.
13	RS.NAL	20000	Region was not previously allocated.
12	RS.NEW	10000	An attach request to a global region was made, and the global region was not found. The global region was created.
11	RS.GBL	4000	Creates a local region within a global region. If global region is not found, returns error. (No error returned if RS.CGR is set.)
10	RS.CGR	2000	Creates a local region within a global region. If a global region is not found, create a global region.
9	RS.AGE	1000	Enables automatic global elimination.
8	RS.EGR	400	Eliminates global region. Global region area is returned to free memory list.
7	RS.EXI	200	Eliminates global region when exiting or aborting from job.
6	RS.CAC	100	Enables cache-bypass.
5	RS.BAS	40	Explicitly assigns base address for this global region.

Table 7–5 (Cont.): Region Status Word (.RDBDF)

Bit	Symbol	Pattern	Meaning When Set
4	RS.NSM	20	Explicitly assigns the base address for this global region exclusive of system memory.
1	RS.DSP	2	Reserved.
0	RS.PVT	1	Reserved.

Table 7–6: Region Control Block (.RCBDF)

Offset	Symbol	Modifier	Contents
0	R.BADD	Monitor's .CRRG routine	The starting address of the region, expressed in 32-word units.
2	R.BSIZ	Monitor's .CRRG routine	The size of the region in 32-word units. If this word is 0, this region control block is free.
4	R.BSTA	The monitor at job initiation; the monitor's .CRRG routine clears this byte	Bit 0 (R.STOP) is set by the monitor. Bit 1 (R.SHAR) indicates that the region is a shared global region. Bit 2 (R.EXIT) indicates that the global region should be eliminated when the job exits or aborts. Bit 3 (R.CACH) indicates the local region is attached to a global region with cache-bypass set.
5	R.BNWD	Monitor's .CRRG routine clears this byte; .MAP increments it; .UNMAP decrements it	The number of windows currently mapped to this region.

Table 7–7: Global Region Control Block (.GRBDF)

Byte Offset	Symbol	Modifier	Contents
0	GR.SIZ	Monitor's .CRRG routine	The size of the global region expressed in 32-word units. A 0 indicates the global region is not in use.
2	GR.ADD	Monitor's .CRRG routine	The starting physical address of the global region expressed in 32-word units.
4	GR.STA	.RDBBK macro	The global region control block status byte.
5	GR.SHC	.RDBBK macro	A count kept of attachments to the global region.
6	GR.NAM	.RDBBK macro	2-word name of global region in Radix–50.

Table 7–8: Global Region Control Block Status Byte, GR.STA

Bit	Symbol	Pattern	Meaning When Set
7	GR.AGE	200	Enables automatic global elimination.
6	GR.PRM	100	The global region is permanent; it is never eliminated.
5	GR.NRF	40	Do not return the global region's allocated memory to the free memory list when the global region is eliminated.
4	GR.CAC	20	Cache-bypass is enabled for the global region.
0-3		Reserved.	

Table 7–9: Window Definition Block (.WDBDF)

Offset	Symbol	Modifier	Contents
0	W.NID	Monitor's .CRAW routine	A unique window identification.

Table 7–9 (Cont.): Window Definition Block (.WDBDF)

Offset	Symbol	Modifier	Contents
1	W.NAPR	.WDBBK macro; monitor's .GMCX routine	The number of the Active Page Register that includes the window's base address.
2	W.NBAS	Monitor's .CRAW and .GMCX routines	The base virtual address of this window; the same address as W.NAPR.
4	W.NSIZ	.WDBBK macro; monitor's .GMCX routine	The size of this window, expressed in 32-word units.
6	W.NRID	.WDBBK macro; monitor's .GMCX routine	Identification of the region to which this window maps.
10	W.NOFF	.WDBBK macro; monitor's .GMCX routine	The offset, expressed in 32_{10}-word units, into the region at which to start mapping this window.
12	W.NLEN	.WDBBK macro; monitor's .MAP and .GMCX routines	The amount of this window to map, expressed in 32-word units. If you put 0 here, .MAP (or.CRAW with WS.MAP set) maps as much of the window as possible.
14	W.NSTS	.WDBBK macro; monitor's .CRAW, .ELAW, and .GMCX routines	The window status word. The .GMCX request clears this word if the window is not mapped.

Table 7–10: Window Status Word, W.NSTS

Bit	Symbol	Pattern	Meaning When Set
0-1	WS.MOD	000003	Processor mode field.
	WS.U	000000	Selected User mode.
	WS.S	000001	Selected Supervisor mode.
	WS.C	000002	Selected current mode.
		000003	Reserved.

Table 7–10 (Cont.): Window Status Word, W.NSTS

Bit	Symbol	Pattern	Meaning When Set
2-3	WS.SPA	000014	Address space field.
	WS.DEF	000000	Default; selected I-space (or D-space if I & D address space separation.)
	WS.I	000004	Selected I-space.
	WS.D	000010	Selected D-space.
		000014	Selected both I-space and D-space.
8	WS.MAP	000400	The .CRAW request should also map the new window in addition to creating it. Set this bit in the window definition block by specifying it in the .WDBBK macro.
9	WS.RO	001000	Access to the addresses mapped by this window is read-only.
10	WS.OVR	002000	Reserved.
11	WS.IDD	004000	The request set WS.I and WS.D and indicated identical action on I-space and D-space windows (the request assumed identical window virtual addresses).
12	WS.DSI	010000	The request set WS.I and WS.D, indicating action on both I-space and D-space. However, D-space is inactive. I-space is processed and D-space is ignored. Not returned by .CRAW request.
13	WS.ELW	020000	The monitor eliminated one or more windows as a result of the current operation. The .CRAW and .ELAW routines can set this bit.
14	WS.UNM	040000	The monitor unmapped one or more windows as a result of the current operation. The .CRAW and .ELAW routines can set this bit. The .MAP and .UNMAP routines set or clear this bit, as required.
15	WS.CRW	100000	The monitor created this window successfully. The .CRAW routine sets this bit; the .ELAW routine clears it.

Table 7–11: Window Control Block (.WCBDF)

Offset	Symbol	Modifier	Contents
0	W.BRCB	Monitor's .MAP routine; the .UNMAP request clears it	A pointer to the region control block of the region to which this window is mapped. If the value is 0, the window is not mapped.
2	W.BLVR	Monitor's .CRAW routine	The window's low virtual address limit.
4	W.BHVR	Monitor's .MAP routine.	The window's high virtual address limit.
6	W.BSIZ	Monitor's .CRAW routine; the .ELAW request clears it	The window's size, in 32_{10}-word units. If the value is 0, this window control block is free.
10	W.BOFF	Monitor's .MAP routine	The offset into the region at which this window begins to map, in 32_{10}-word units.
12	W.BFPD	Monitor's .CRAW routine	The low byte of the address of the first (and therefore the User I-space) Page Descriptor Register that affects this window.
13	W.BNPD	Monitor's .MAP routine	The number of Page Descriptor Registers that affect this window.
14	W.BLPD	Monitor's .MAP routine	The contents of the last Page Descriptor Register that affects this window.

Table 7–12: System Communication Area, .SYCDF

Location	Symbol	Contents
40,41	$USRPC	Start address of job. When you link a file to create an RT–11 executable image, the linker sets the word at address 40 in the program's file to the starting address of the program.
44,45	$JSW	Job Status Word (JSW). This is a flag word for the monitor. The monitor maintains some of the bits itself, and your program can set or clear others.

Table 7–12 (Cont.): System Communication Area, .SYCDF

Location	Symbol	Contents
46,47	$UFLOA	USR load address. This word is normally 0, but you can set it in the file or at run time to any valid word address in your program. If this word is 0, the USR loads in its default location through an address contained in offset 266 ($USRLC of .FIXDF) of RMON.
50,51	$USRTO	High memory address. In this word the monitor maintains the highest address your program can use.
52	$ERRBY	EMT error code. If a monitor request results in an error, the code number of the error is always returned in byte 52 in memory and the carry bit is set.
53	$USRRB	User program error code (USERRB). If a user program encounters errors during execution, it indicates the error by using this byte in memory.
54,55	$SYPTR	Address of the beginning of RMON. RT–11 always loads the monitor into the highest available memory locations of low (rather than extended) memory.
56	$TTFIL	Fill character (seven-bit ASCII). Some very old terminals require fill (null) characters after printing certain characters.
57	$TTNFI	Fill count. This byte in memory specifies the number of fill characters that are required. The number of characters is determined by hardware. If bytes 57 and 56 are 0, no fill is required.

Table 7–13: User Error Byte, .UEBDF

Bit	Mask	Symbol	Status	RT–11 Message
0	1	SUCCS$	Success	?PROG-I-text, or none
1	2	WARN$	Warning	?PROG-W-text
2	4	ERROR$	Error	?PROG-E-text
3	10	FATAL$	Fatal	?PROG-F-text
4	20	UNCON$	Unconditional	?PROG-U-text

Table 7–14: Job Definition Word ($JSX) Bit Definitions

Bit Mask	Symbol	Meaning
000001	REQID$	Job requires separated I–D space.
000002	USEID$	Job uses separated I–D space, if possible.
000004	REQSM$	Job requires Supervisor mode support.
000010	USESM$	Job uses Supervisor mode, if possible.
000020	ALL64$	.SETTOP under VBGEXE supported to top of memory (-2).
000040	IOPAG$	IOPAGE region under VBGEXE is mapped to PAR7.
000100	NOVBG$	Job cannot be run under VBGEXE.
000200	VBGEX$	Job can be run under VBGEXE.
000400-100000		Reserved.

Table 7–15: Block-Replaceable Device Directory Structure

Header:		
Name	**Offset**	**Contents**
D.TOTA	0	Total number of segments in this directory; the valid range is 1 to 31(decimal)
D.NEXT	2	The segment number of the next logical directory segment; 0 if none
D.HIGH	4	The number of the highest segment currently in use; only maintained in first directory segment
D.EXTR	6	The number of extra bytes per directory entry; always an unsigned, even octal number
D.STRT	10	The block number on the device where the actual stored data monitored by this segment begins
Directory Entry:		
Name	**Offset**	**Contents**
–	0	Entry status word
E.NAME	2	File name; three words of Radix–50
E.LENG	10	Size in blocks of file, or size of free area if unused or tentative
E.USED	12	Unused on disk for permanent files; in memory, highest block written
E.CHAN	12	Channel number of tentative entry (byte)
E.JNUM	13	FB, XM; Job number of tentative entry (byte)
E.DATE	14	Creation date
	16+	Extra words, if any

Table 7–16: Home Block Contents

Field	Location	Contents	Default
a	$000–201_8$	Bad block replacement table	
b	$204–251_8$	INITIALIZE/RESTORE data area	
c	$252–273_8$	BUP information area	If BUP volume: "BUQ" and 9 spaces; binary volume number in byte 266_8
d	$700–701_8$	(Reserved for Digital)	000000
e	$702–703_8$	(Reserved for Digital)	000000
f	$722–723_8$	Pack cluster size	000001
g	$724–725_8$	Block number of first directory segment	000006
h	$726–727_8$	System version	Radix–50 "V3A"
i	$730–743_8$	Volume Identification	"RT11A" and seven spaces
j	$744–757_8$	Owner name	12_{10} spaces
k	$760–773_8$	System Identification	"DECRT11A" and four spaces
l	$776–777_8$	Checksum	

Table 7–17: Bad Block Replacement Table Entries

Offset	Contents
0	Bad block number; 0 if end of table
2	Replacement block number

Table 7–18: .SAV Image File (Block 0)

Offset	Contents
0	VIR in Radix–50 if the Linker /V option was used.
2	Virtual high limit if Linker /V option was used.
4	Job definition word ($JSX) bits.
6	Reserved
10	Reserved
12	Reserved
14	BPT trap PC (mapped monitors only)
16	BPT trap PSW (mapped monitors only)
20	IOT trap PC (mapped monitors only)
22	IOT trap PSW (mapped monitors only)
24	Reserved
26	Reserved
30	Reserved
32	Overlay definition word (SV.CVH) bits.
34	Trap vector PC (TRAP)
36	Trap vector PSW (TRAP)
40	Program's relative start address
42	Initial location of stack pointer (changed by /M option)
44	Job Status Word
46	USR swap address
50	Program's high limit
52-62	Reserved
64	Address of overlay handler table for overlaid files
66	Address of start of window definition blocks (if /V used)

Table 7–18 (Cont.): .SAV Image File (Block 0)

Offset	Contents
70–356	Reserved
360–377	Bitmap area

Table 7–19: SET Options Table

Offset	Name	Meaning
0	DSE.R3	Value to pass in R3 to the SET routine
2–4	DSE.NA	Radix–50 for option name (two words)
6	DSE.SB	Offset to option routine
7	DSE.PA	Parsing option bits:

Bit	Name	Meaning
0–4		Reserved
5	DSE.8	Set means option has octal value Clear means option has decimal value
6	DSE.NU	Numeric value allowed
7	DSE.NO	No prefix allowed

–	DSE.ES	Entry size

Table 7–20: SET Options Status Word ($TTCNF)

Bit	Symbol	Meaning When Set
0	HWTAB$	SET TT: TAB option is in effect.
1	CRLF$	SET TT: CRLF option is in effect.
2	FORM$	SET TT: FORM option is in effect.
3	FBTTY$	SET TT: FB option is in effect.
4-6		Reserved
7	PAGE$	SET TT: PAGE option is in effect.
8-14		Reserved.

Table 7–20 (Cont.): SET Options Status Word ($TTCNF)

Bit	Symbol	Meaning When Set
15	BKSP$	SET TT: SCOPE option is in effect.

Table 7–21: RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
0	$RMON	4	Common interrupt entry point; contains the instruction JMP $INTEN. The .INTEN macro uses it.
4	$CSW	240	Background job channel area (16_{10}) channels; each is five words long.
244	$SYSCH	12	Internal channel used for system functions; KMON uses this channel.
246-254		2	Reserved.
256	$BLKEY	2	Segment number of the directory now in memory. A value of 0 implies that no directory is there.
260	$CHKEY	2	Device index and unit number of the device whose directory is in memory. The low byte contains the device index into the monitor tables; the high byte is the unit number.
262	$DATE	2	Current date value.
264	$DFLG	2	''Directory operation in progress'' flag. This is non-zero to inhibit CTRL/C from aborting a job while a directory operation is in progress.
266	$USRLC	2	Address of the normal USR area. This is where the USR resides when it is called into memory by the background job and $UFLOA (location 46) is 0.
270	$QCOMP	2	Address of the I/O exit routine for all devices.

Table 7–21 (Cont.): RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
272	SPUSR	2	Special device error word. Non-RT–11 file-structured devices, such as magtape, use this word to report errors to the monitor.
274	$SYUNI	2	The high byte contains the unit number of the system device. This is the unit number of the device from which the system was bootstrapped.
276	$SYSVE	1	Monitor version number. You can always access the version number in this fixed offset to determine if you are using the most recent version of the software.
277	$SYSUP	1	Monitor release level. This number identifies the release level of the monitor version specified in byte 276.
300	$CNFG1	2	Configuration word. These 16 bits indicate information about either the hardware configuration of the system or a software condition.
302		2	Reserved. (Obsolete; always 0.)
304	$TTKS	2	Address of the console keyboard status register. The default value is 177560.
306	$TTKB	2	Address of the console keyboard buffer register. The default value is 177562.
310	$TTPS	2	Address of the console printer status register. The default value is 177564.
312	$TTPB	2	Address of the console printer buffer register. The default value is 177566.
314	$MAXBL	2	The maximum file size allowed in a 0 length .ENTER programmed request. The default value is 177777_8 blocks, allowing an essentially unlimited file size.
316	$E16LS	2	Offset from the start of RMON to the dispatch table for EMTs 340 through 357.

Table 7–21 (Cont.): RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
320	$CNTXT	2	A pointer to the impure area for the current executing job. (Defined by .IMPDF macro in SYSTEM.MLB)
322	$JOBNU	2	The executing job's number.
324	$SYNCH	2	Address of monitor routine to handle .SYNCH requests.
326	$LOWMA	24	Start of the low-memory protection map. This map protects vectors at locations 0 through 476.
352	$USRLO	2	A pointer to the current entry point of the USR.
354		2	Reserved. (Obsolete; always 0.)
356	$ERRCN	2	Low byte is the error count byte for use by system utility programs. The high byte is reserved.
360	$MTPS	2	Entry point of the move to PS routine. The .MTPS macro calls this routine to perform processor independent moves to the Processor Status word.
362	$MFPS	2	Entry point of the move from PS routine. The .MFPS macro calls this routine to do processor independent moves from the Processor Status word.
364	$SYIND	2	Index into the monitor device tables for the system device.
366	$STATW	2	Indirect file and monitor command state word.
370	$CNFG2	2	Extension configuration word. This is a string of 16 bits indicating the presence of an additional set of hardware options on the system.
372	$SYSGE	2	System generation features word. The bits in this word indicate the presence or absence of some system generation special features.

Table 7–21 (Cont.): RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
374	$USRAR	2	Size of the USR in bytes. Your program can use this information to dynamically determine the size of the region you need in order to swap the USR. (The USR is always resident in mapped systems.)
376	$ERRLE	1	Error severity at which to abort indirect command files.
377	$IFMXN	1	Depth of nesting of indirect files. The default nesting level is 3.
400	$EMTRT	2	Internal offset for use by SL and BATCH only.
402	$FORK	2	Offset to fork processor from the start of RMON. Use the .DREND macro in your device handler to automatically set up a pointer to the fork processor.
404	$PNPTR	2	Offset to the $PNAME table from the start of RMON.
406	$MONAM	4	Two words of Radix–50 containing the name of the current monitor file.
412	$HSUFF	2	One word of Radix–50 containing the suffix used by the current monitor to name device handlers. For unmapped systems, this word is normally blank. For mapped systems, it is normally *X*, right-justified.
414	$SPSTA	2	Status of the transparent spooler.
416	$EXTIN	1	IND stored error byte. (Defined by .UEBDF macro in SYSTEM.MLB)
417	$INDST	1	IND control status byte.
420	$MEMSZ	2	Total physical memory available, in 32-word blocks.
422	$ELTIM	2	Reserved for error logger.
424	$TCFIG	2	Address of terminal SET option status word. (Defined by .TTCDF macro in SYSTEM.MLB)

Table 7–21 (Cont.): RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
426	$INDDV	2	Pointer to INDDEV (the ASCII name of the device from which IND was run). Also pointer to a 2-byte KMON/UCF interface, located in the first word under INDDEV.
430	$MEMPT	2	Offset to memory control block pointers. (Defined by .MEMDF macro in SYSTEM.MLB)
432	P1$EXT	2	Pointer to $P1EXT routine.
434-444		2	Reserved.
446	$IMPLO	2	Pointer to word following last word in table of pointers to the impure area.
450	$KMONI	2	Indicates if KMON is the background job. If $KMONI contains nonzero value, then KMON is background job; if $KMONI contains zero, then KMON not background job.
452	$PROGD	1	Indicates default editor for EDIT command.
453	$PROGF	1	Indicates which FORTRAN compiler is run by the COMPILE/FORTRAN, EXECUTE/FORTRAN, and FORTRAN commands.
454	$WILDD	1	Indicates if wildcarding (*) is implicit or must be supplied explicitly. A value of one indicates implicit wildcarding; zero indicates explicit wildcarding.
455	$JOBS	1	Indicates the number of job slots available on the system.
456	$QHOOK	2	Pointer to RMON code hooks for UB system support handler. (Defined by .QHKDF macro in SYSTEM.MLB)
460	$H2UB	2	Pointer to the UB system support handler entry vector.

Table 7–21 (Cont.): RMON Fixed Offsets

Offset	Symbol	Byte Length in Octal	Description
462	$XOFF	2	Pointer/offset to an XON/XOFF flag. In nonmultiterminal monitors, $XOFF points to a byte that contains the XON/XOFF flag. In multiterminal monitors, $XOFF contains the offset in the TCB to the T.STAT word.
464	$RTSPC	4	Address of SEC/RETURN instructions.
466	$CNFG3	2	Third configuration word.
470		2	Reserved.
472	$THKPT	2	Pointer to the optional multiterminal handler hooks data structure, THOOKS.
474		2	Reserved. (Obsolete; always 0.)
476	$XTTPS	2	Terminal service calls @$XTTPS before reading @$TTPS and after changing @$TTPS.
500	$XTTPB	2	Terminal service calls @$TTPB after changing @$TTPB.
502	$SLOT2	1	The value of $SLOT*2.
503		1	Reserved.
504	$SPSIZ	2	Special device file size. Used to return value from a special directory handler to the monitor. The monitor returns this value in R0 for special directory operations

Table 7–22: $CNFG1 Configuration Word, Offset 300 (.CFIDF)

Bit	Symbol	Meaning
0	FBMON$	0 = SET MODE SJ in effect. 1 = SET MODE NOSJ in effect.
1	SLKMO$	1 = KMON uses SL editor.
2		Reserved. (Obsolete; always 0.)
3	BATCH$	1 = BATCH is in control of the background.
4	SLEDI$	1 = Single-line editor is available to user programs.
5	CLK50$	0 = 60-cycle clock. 1 = 50-cycle clock. The value can be controlled by SET CLOCK 50/60.
6	HWFPU$	1 = FP11 floating-point hardware exists.
7	FJOB$	0 = No foreground or system job is in memory. 1 = A foreground or system job is in memory.
8		Reserved. (Obsolete; always 0.)
9	USR$	1 = USR is permanently resident, via SET USR NOSWAP. (USR is always resident in mapped systems and this bit is always set.)
10	QUEUE$	1 = The QUEUE program is running.
11	LSI11$	1 = The Processor Status word on this system cannot be accessed by means of an address in the I/O page.
12	KT11$	1 = A mapped system is running.
13	LKCS$	1 = The system clock has a status register.
14	KW11P$	1 = A KW11-P clock exists and programs can use it.
15	CLOCK$	1 = There is a system clock (L clock, P clock, or 11/03-11/23 line-frequency clock).

Table 7–23: DCL and IND File Status Word, $STATW (.STWDF)

Bit Mask	Symbol	Meaning
000001- 000002		Reserved.
000004	IFIND$	0 = @ means indirect command file 1 = @ means indirect control file
000010	IFDOL$	Dollar sign ($) entered at command line
000020	IFSPC$	Special chain exit
000040	IFBEX$	BATCH is forcing exit after error in user job
000100	IFRVT$	Revert to TTY input from indirect file
000200	IFGTC$	CTRL/C seen in indirect file while GTLIN$ bit set in $JSW
000400	IFACT$	Indirect file active
001000	IFCHA$	Chain to indirect file
002000	IFEKO$	Don't echo indirect file lines
004000	IFCTC$	CTRL/C seen in indirect file
010000	IFDAT$	Data in indirect file buffer above the USR
020000	IFEOF$	EOF in indirect command file
040000	IFABT$	Abort indirect command file input
100000	IFINP$	Input from indirect command file

Table 7–24: $CNFG2 Extension Configuration Word, Offset 370 (.CF2DF)

Bit	Symbol	Meaning
0	CACHE$	1 = Cache memory is present.
1	MPTY$	1 = Parity/ECC memory is present.
2	SWREG$	1 = A readable switch register is present.
3	LIGHT$	1 = A writeable console display register is present.
4	LDREL$	1 = A handler used by LD may have been unloaded.

Table 7–24 (Cont.): $CNFG2 Extension Configuration Word, Offset 370 (.CF2DF)

Bit	Symbol	Meaning
5	XITSW$	1 = Do not swap user code or exit.
6	BUS$	See below.
7	CIS$	1 = The Commercial Instruction Set (CIS) option is present.
8	EIS$	1 = The Extended Instruction Set (EIS) option is present.
9-10		Reserved. (Obsolete; always 0.)
11	KXCPU$	See below.
12	GSCCA$	1 = Global SCCA support in monitor.
13	PROS$	See below.
14	PDP70$	1 = The processor is a PDP–11/70.
15	PDP60$	1 = The processor is a PDP–11/60.

Table 7–25: System Generation Features Word, Offset 372 (.SGNDF)

Bit	Symbol	Meaning
0	ERLG$	1 = The error logging feature is present.
1	MMGT$	1 = The memory management feature is present.
2	TIMIT$	1 = The device I/O time-out feature is present.
3	RTEM$	1 = This is an RTEM-11 system.
4-7		Reserved.
8	FPU11$	1 = FPU support selected during system generation.
9	MPTYS$	1 = The memory parity feature is present.
10	TIMER$	1 = The SB mark time feature is present.
11-12		Reserved.
13	MTTY$	1 = The multiterminal feature is present.
14	STASK$	1 = The system job feature is present.

Table 7–25 (Cont.): System Generation Features Word, Offset 372 (.SGNDF)

Bit	Symbol	Meaning
15	TSXP$	Running under an operating system other than RT–11.

Table 7–26: Transparent Spooler Status Word, Offset 414 (.SPLDF)

Bit	Symbol	Meaning
0-2	SP$UNI†	Unit number for SET commands. Meaning of unit number bits tied to SP$OPR. See note.
3-6	SP$OPR	Type of operation to perform.
7	SP$ACT	Spooler active.
8-10	SP$NFL	Type of flag page support.
11	SP$SHO	1 = Display spooler status.
12	SP$SCN	1 = Print screen (CTI bus-based computers only).
13	SP$GTM	1 = Date and time request for flag pages.
14	SP$IEN	1 = Fake interrupt enable.
15	SP$ERR	1 = Error bit (set by SPOOL).

†Meaning of spooler unit field (SP$UNI) is determined by a bit mask:

- If SP$OPR is SP$NXT, SP$OFF, SP$ON, or SP$KIL, then SP$UNI is the SP unit number.
- If SP$OPR is SP$ABT, then SP$UNI is the aborting job number.
- If SP$OPR is SP$EXI, then SP$UNI has no meaning.

Table 7–27: IND Control Status Byte, Offset 417 (.INDDF)

Value	Symbol	Meaning
001		Reserved
002		Reserved
004	CC$IND	Set if double CTRL/C abort is disabled by IND.

Table 7–27 (Cont.): IND Control Status Byte, Offset 417 (.INDDF)

Value	Symbol	Meaning
010	CC$GBL	Set if double CTRL/C is disabled by global .SCCA
040	LN$IND	Set if current line passed by IND
100	IN$RUN	Set if KMON issued RUN of IND
200	IN$IND	Set if IND active

Table 7–28: Third Configuration Word, Offset 466 (.CF3DF)

Value	Symbol	Meaning
000004	CF3.IM	Each job possesses an impure area.
000010	CF3.VB	RUN is SET to VBGEXE.
000020	CF3.UI	The UB system support handler is set to not install.
000040	CF3.UA	The UB system support handler is active (set only when CF3.UB is set).
000100	CF3.UB	The UB system support handler is loaded in memory.
000200	CF3.DM	At least one handler installed on the system uses DMA (direct memory access).
000400	CF3.64	RMON has been generated for extended device-unit support and the $PNAM2 table exists in RMON.
001000	CF3.AT	$JBREL exists in monitor.
002000	CF3.OW	$OWNER table exists in RMON.
004000	CF3.US	Support for UB system support handler in the PNAME table from BSTRAP.
010000	CF3.1S	RMON supports exactly one DL11 console interface.
020000	CF3.AS	Multiterminal monitor supports asynchronous terminal status (AST) word.
040000	CF3.HI	The hardware supports separated I & D address space and Supervisor Mode.

Table 7–28 (Cont.): Third Configuration Word, Offset 466 (.CF3DF)

Value	Symbol	Meaning
100000	CF3.SI	The monitor supports separated I & D address space and Supervisor Mode.

Table 7–29: Impure Area (.IMPDF)

Offset	Symbol	Byte Length (Octal)	Description
0	I.STATE	2	Job state word bits.
2	I.QHDR	2	Head of available queue element linked list.
4	I.CMPE	2	Last entry in the completion queue.
6	I.CMPL	2	Head of the completion queue.
10	I.CHWT	2	Pointer to channel during I/O wait. When a job is waiting for I/O on a channel to complete, the address of that channel area is stored here.
12	I.PCHW	2	Saved I.CHWT during execution of a completion routine.
14	I.PERR	2	Error bytes 52 and 53 saved during completion routines.
16	I.TTLC	2	Terminal input ring buffer line count (for non-multiterminal systems).
16	I.CNSL	2	Multiterminals only: Pointer to terminal control block (TCB) for this job's console terminal.
20	I.PTTI	2	Previous terminal input character (for non-multiterminal systems).
20	unused	2	Multiterminals only: reserved.
22	I.TID	2	Pointer to job ID area, later in impure area.
24	I.JNUM	2	Job number of the job that owns this impure area.

Table 7–29 (Cont.): Impure Area (.IMPDF)

Offset	Symbol	Byte Length (Octal)	Description
26	I.CNUM	2	Number of I/O channels defined. The default is 16_{10}; you can use .CDFN to define more.
30	I.CSW	2	Pointer to the job's channel area.
32	I.IOCT	2	Total number of I/O operations outstanding.
34	I.SCTR	2	Suspension counter. A value less than 0 means the job is suspended.
36	I.BLOK	2	Job blocking bits.
-	I.JID	10	Job's terminal prompt string. If the system job feature is present, the length of I.JID is 14_8.
-	I.LNAM	6	System jobs only: Logical job name in ASCII.
-	I.NAME	10	File name and file type, in Radix–50, of the running job.
-	I.SPLS	2	Pointer to nonlinked .DEVICE list.
-	I.TRAP	2	Address of trap to 4 and 10 routine defined via .TRPSET.
-	I.FPP	2	FPU only: Address of FPP exception routine defined via .SFPA.
-	I.SPSV	2	Mapped only: Bottom of saved SP data.
-	I.SWAP	4	Multi-job only: Pointer to extra swap information specified in the .CNTXSW programmed request.
-	I.SP	2	Multi-job only: Saved stack pointer.
-	I.BITM	24	Multi-job only: Bitmap for protection.
-	I.CLUN	2	Multiterminals only: LUN of job's console.
-	I.TTLC	2	Multiterminals only: Terminal input ring buffer line count.
-	I.IRNG	2	Input ring buffer low limit.

Table 7–29 (Cont.): Impure Area (.IMPDF)

Offset	Symbol	Byte Length (Octal)	Description
-	I.IPUT	2	Input PUT pointer for interrupts.
-	I.ICTR	2	Input character count.
-	I.IGET	2	Input GET pointer for .TTYIN.
-	I.ITOP	2	Input ring buffer high limit.
-	——	TTYIN	Input ring buffer.
-	I.OPUT	2	Output PUT pointer for .TTYOUT.
-	I.OCTR	1	Output character count.
-	I.CTLO	1	^C flag
-	I.OGET	2	Output GET pointer for interrupts.
-	I.OTOP	2	Output ring buffer high limit.
-	——	TTYOUT	Output ring buffer.
-	I.QUE	QWDSIZ	The initial queue element; 16_8 bytes (24 bytes if mapped).
-	——	4	Multi-job only: First internal message channel words.
-	I.SERR	2	(All monitors) The third word of the message channel is used as the hard/soft error flag.
-	I.MSG	4	Multi-job only: Further internal message channel words.
-	I.TERM	2	Terminal status word. (.TSTDF)
-	I.TRM2	2	Terminal status word 2. (.TS2DF)
-	I.SCCA	2	CTRL/C terminal status word set via .SCCA. (.TASDF)
-	I.SCC1	2	Mapped only: PAR1 value of I.SCCA.
-	I.DEVL	2	Pointer to linked .DEVICE list.

Table 7–29 (Cont.): Impure Area (.IMPDF)

Offset	Symbol	Byte Length (Octal)	Description
-	I.FPSA	2	Mapped and FPU only: Pointer to FPU save area, later in impure area.
-	I.SCOM	36	Mapped only: System communication save area (for non-multiterminal systems).
-	I.SCOM	40	Mapped and multiterminals only: System communication save area.
-	I.RSAV	20	Mapped only: Register save area.
-	I.MPTR	2	Mapped only; Pointer to job's mapping context area (MCA) in extended memory. The PAR1 value.
-	I.MPB	16	Mapped only; Temporary copy of the job's parameter block for a PLAS request; contains an RDB or WDB, depending on type of request.
-	I.PLSP	2	Fully mapped (ZB or ZM) only; D-space / I-space 'pass' parameter for PLAS requests.
-	I.CMAP	2	Fully mapped (ZB or ZM) only; .CMAP request status.
-	I.SSP	2	Fully mapped (ZB or ZM) only; saved Supervisor Mode stack pointer.
-	I.FSAV	62	Mapped and FPU only: FPU save area.
-	I.VHI	2	Mapped only: Virtual high limit of job; nonzero if linker /V option used.
-	I.VSTP	2	Mapped only; used for nonvirtual .SETTOP; maximum high limit for completely virtual jobs.
-	I.ECTR	2	.SPCPS only: EMT depth counter.
-	I.SPCP	2	.SPCPS only: Address of .SPCPS blocks.
-	I.SPC1	2	Mapped and .SPCPS only: PAR1 for .SPCPS blocks.

Table 7–29 (Cont.): Impure Area (.IMPDF)

Offset	Symbol	Byte Length (Octal)	Description
-	I.SCHP	2	Pointer to the job's system channel. The monitor uses this channel for its own calls, such as .DSTATUS.
-	I.SYCH	14	The job's system channel, for all foreground and system jobs. The background job's channel is in the fixed offset area of RMON.
-	IMP.SZ	0	Symbol whose value is the length in bytes of impure area.

Table 7–30: Job State Word Bits, Offset 0 (.ISTDF)

Mnemonic	Bit	Meaning When Set
ABPND$	0	An abort has been requested for this job.
BATRN$	1	BATCH or Error Logger is running for this job.
CSIRN$	2	The CSI is running for this job.
USRRN$	3	The USR is running for this job.
-	4	Reserved.
ABORT$	5	The job is being aborted.
-	6	Reserved.
CPEND$	7	This job has a completion routine pending.
-	8-10	Reserved.
LOAD$	11	Mapped only; foreground or system job is in the load phase; the job has not yet started running. Context switch code in RMON will initialize mapping and clear LOAD$ bit.
WINDW$	12	This is a virtual job.
VRUN$	13	This job is running under VBGEXE.
VLOAD$	14	This job is being loaded under VBGEXE.

Table 7–30 (Cont.): Job State Word Bits, Offset 0 (.ISTDF)

Mnemonic	Bit	Meaning When Set
CMPLT$	15	A completion routine is running for this job.

Table 7–31: Job Blocking Bits, Offset 36 (.IBKDF)

Mnemonic	Bit	Meaning When Set
-	0-3	Reserved.
USRWT$	4	The job is waiting for the USR.
-	5	Reserved.
KSPND$	6	The job is suspended as a result of the SUSPEND command.
-	7	Reserved.
EXIT$	8	The job is waiting for all I/O to complete.
NORUN$	9	The job is not running (that is, it is a foreground or system job)
SPND$	10	The job is suspended.
CHNWT$	11	The job is waiting for I/O on a channel to complete.
TTOEM$	12	The job is waiting for the output ring buffer to be empty.
TTOWT$	13	The job is waiting for room in the output ring buffer.
TTIWT$	14	The job is waiting for terminal input.
-	15	Reserved.

Table 7–32: Spool Operation Type Field (SP$OPR)

Bit Mask	Symbol	Meaning
0001	SP$NXT	Move to start of next file.
0010	SP$OFF	Set spooler unit off.
0100	SP$ON	Set spooler unit on.

Table 7–32 (Cont.): Spool Operation Type Field (SP$OPR)

Bit Mask	Symbol	Meaning
1000	SP$KIL	Remove spooled output from work file.
0110	SP$ABT	Spooler aborted.
1110	SP$EXI	Cause spooler shutdown.

Table 7–33: Spool Flag Page Support Field (SP$NFL)

Bit Mask	Symbol	Meaning
000	SP$DEF	Default number of flagpages.
111	SP$0	No flagpages.
xxx		This number of flagpages, where *xxx* is in the range 1-6.

Table 7–34: Change Mapping Context (I.CMAP) Word Bits (.CMPDF)

Bit Mask	Symbol	Meaning
000001	CM.PR0	Separate PAR0 mapping.
000002	CM.PR1	Separate PAR1 mapping.
000004	CM.PR2	Separate PAR2 mapping.
000010	CM.PR3	Separate PAR3 mapping.
000020	CM.PR4	Separate PAR4 mapping.
000040	CM.PR5	Separate PAR5 mapping.
000100	CM.PR6	Separate PAR6 mapping.
000200	CM.PR7	Separate PAR7 mapping.
001400	CM.S	Supervisor mode I & D Separation Field.
001000	CM.SII	Non-separate Supervisor I & D space.

Table 7–34 (Cont.): Change Mapping Context (I.CMAP) Word Bits (.CMPDF)

Bit Mask	Symbol	Meaning
001400	CM.SID	Separate Supervisor I & D space.
002000		Reserved.
030000	CM.SUP	Supervisor mode context switching support field.
020000	CM.NOS	No Supervisor mode context switching.
030000	CM.JAS	Supervisor mode context switching.
140000	CM.U	User mode I & D separation field.
100000	CM.UII	Non-separate User I & D space.
140000	CM.UID	Separate User I & D space.

Table 7–35: MCA Region Structure (.MCADF)

Symbol	Contents
M.RGN	Base of the MCA region and location of the RCBs. Can contain up to 24_{10} RCBs.
M.APR	Beginning of APR (Active Page Register) section. APRs consist of PDRs (Page Descriptor Registers) and PARs (Page Address Registers). Individual APRs can be mapped by specifying the CM.DUS and CM.PAR arguments in the .CMAP request. Alternately, the .MSDS request can map individual APRs after general support has been established by the .CMAP request.
M.PDUI	User mode/Instruction space PDRs (8 words). These PDRs are present in all mapped systems and are always context-switched.

Table 7–35 (Cont.): MCA Region Structure (.MCADF)

Symbol	Contents
M.PDUD	User mode/Data space PDRs (8 words). These PDRs are context-switched only when CM.UID has been specified in a .CMAP request. With single-mapped monitors, this space is undefined but reserved.
M.PAUI	User mode/Instruction space PARs (8 words). These PARs are present in all mapped systems and are always context-switched.
M.PAUD	User mode/Data space PARs (8 words). These PARs are context-switched only when CM.UID has been specified in a .CMAP request. With single-mapped monitors, this space is undefined but reserved.
M.PDSI	Supervisor mode/Instruction space PDRs (8 words). These PDRs are context-switched only when CM.JAS has been specified in a .CMAP request. Fully-mapped monitors only.
M.PDSD	Supervisor mode/Data space PDRs (8 words). These PDRs are context-switched only when CM.JAS and CM.SID have been specified in a .CMAP request. Fully-mapped monitors only.
M.PASI	Supervisor mode/Instruction space PARs (8 words). These PARs are context-switched only when CM.JAS has been specified in a .CMAP request. Fully-mapped monitors only.
M.PASD	Supervisor mode/Data space PARs (8 words). These PARs are context-switched only when CM.JAS and CM.SID have been specified in a .CMAP request. Fully-mapped monitors only.
M.WCB	Beginning of the WCB section. Up to 8 WCBs are supported for each processor mode/address space. WCBs are maintained in each processor mode/address space configuration in alignment with the corresponding APR (the APR that the WCB describes).
W.WCUI	User mode/Instruction space WCBs. User mode/Instruction space WCBs are active for all mapped environments.
W.WCUD	User mode/Data space WCBs.
W.WCSI	Supervisor mode/Instruction space WCBs.
W.WCSD	Supervisor mode/Data space WCBs.

Table 7–35 (Cont.): MCA Region Structure (.MCADF)

Symbol	Contents
M.MMR3	Memory mapping register 3. Do not modify the contents of M.MMR3. The relevant bits for mapping context are:

Bit	Meaning When Set
0	Separate User mode I-D address space.
1	Separate Supervisor mode I-D address space.
3	The CSM instruction is enabled.

Queue Elements

Table 7–36: Completion Queue Element Format (.QCMDF)

Name	Offset	Contents
QC.LNK	0	Link to next queue element; 0 if none
	2-6	Reserved
QC.CSW	10	Channel Status Word
QC.OFT	12	Offset from start of channel area to this channel
QC.CMP	14	Completion routine address

Table 7–37: Fork Queue Element Format

Offset	Name	Agent	Contents
0	F.BLNK	monitor	Link word
2	F.BADR	monitor	Address of FORK routine
4	F.BR5	monitor	R5 save area
6	F.BR4	monitor	R4 save area

Table 7–38: I/O Queue Element Format

Name	Offset	Contents
Q.LINK	0	Link to next queue element; 0 if none
Q.CSW	2	Pointer to Channel Status Word in I/O channel
Q.BLKN	4	Physical block number

Table 7–38 (Cont.): I/O Queue Element Format

Name	Offset	Contents
	6	Special function support; contents determined by device-unit support: If 8-unit handler: Q.FUNC Function code (8 bits) If Extended Device-Unit handler: Q.FUNC 4 low bits for special function Q.2UNI 3 bits (MSB) of unit number Q.TYPE 1 bit when set means special function
	7	Unit and job number: Q.UNIT 3 bits (LSB) of unit number Q.JNUM 4 bits job number; high bit reserved and always 0
Q.BUFF	10	User buffer address (mapped through PAR1 with Q.PAR or Q.MEM value, if mapped monitor)
Q.WCNT	12	Word count: If < 0, operation is WRITE If = 0, operation is SEEK If > 0, operation is READ The true word count is the absolute value of this word.
Q.COMP	14	Completion routine code: If 0, this is wait-mode I/O If 1, just queue the request and return If even, this is a completion routine address
Q.ELGH	16	Size of I/O queue element if not mapped
Q.PAR	16	UMR relocation constant (mapped monitors only)
Q.MEM	20	PAR1 displacement bias for MMU (mapped monitors only)
	22	Reserved (mapped monitors only)
Q.ELGH	24	Size of I/O queue element if mapped

Table 7–39: Synch Queue Element (.QSYDF)

Name	Offset	Contents
QS.LNK	0	Link to next queue element; 0 if none
QS.JOB	2	Job number
	4-6	Reserved
QS.ID	10	Synch ID
QS.SYN	12	-1 (cue that this is a synch element)
QS.CMP	14	Synch routine address

Table 7–40: Timer Queue Element Format (.QTIDF)

Name	Offset	Contents
C.HOT	0	High-order time
C.LOT	2	Low-order time
C.LINK	4	Link to next queue element; 0 if none
C.JNUM	6	Owner's job number
C.SEQ	10	Owner's sequence number ID
C.SYS	12	-1 if system timer element; -3 if .TWAIT element in mapped system
C.COMP	14	Address of completion routine

Chapter 8

Standard References

Table 8–1: ASCII Character Set, Left/Right Byte Equivalents 000-177

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
000000	000	NUL	0	020000	040	SP	32
000400	001	SOH	1	020400	041	!	33
001000	002	STX	2	021000	042	"	34
001400	003	ETX	3	021400	043	#	35
002000	004	EOT	4	022000	044	$	36
002400	005	ENQ	5	022400	045	%	37
003000	006	ACK	6	023000	046	&	38
003400	007	BEL	7	023400	047	’	39
004000	010	BS	8	024000	050	(	40
004400	011	HT	9	024400	051	)	41
005000	012	LF	10	025000	052	*	42
005400	013	VT	11	025400	053	+	43
006000	014	FF	12	026000	054	,	44
006400	015	CR	13	026400	055	—	45
007000	016	SO	14	027000	056	.	46
007400	017	SI	15	027400	057	/	47
010000	020	DLE	16	030000	060	0	48

Table 8–1 (Cont.): ASCII Character Set, Left/Right Byte Equivalents 000-177

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
010400	021	DC1	17	030400	061	1	49
011000	022	DC2	18	031000	062	2	50
011400	023	DC3	19	031400	063	3	51
012000	024	DC4	20	032000	064	4	52
012400	025	NAK	21	032400	065	5	53
013000	026	SYN	22	033000	066	6	54
013400	027	ETB	23	033400	067	7	55
014000	030	CAN	24	034000	070	8	56
014400	031	EM	25	034400	071	9	57
015000	032	SUB	26	035000	072	:	58
015400	033	ESC	27	035400	073	;	59
016000	034	FS	28	036000	074	<	60
016400	035	GS	29	036400	075	=	61
017000	036	RS	30	037000	076	>	62
017400	037	US	31	037400	077	?	63
040000	100	&	64	060000	140	`	96
040400	101	A	65	060400	141	a	97
041000	102	B	66	061000	142	b	98
041400	103	C	67	061400	143	c	99
042000	104	D	68	062000	144	d	100
042400	105	E	69	062400	145	e	101
043000	106	F	70	063000	146	f	102
043400	107	G	71	063400	147	g	103

Table 8–1 (Cont.): ASCII Character Set, Left/Right Byte Equivalents 000-177

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
044000	110	H	72	064000	150	h	104
044400	111	I	73	064400	151	i	105
045000	112	J	74	065000	152	j	106
045400	113	K	75	065400	153	k	107
046000	114	L	76	066000	154	l	108
046400	115	M	77	066400	155	m	109
047000	116	N	78	067000	156	n	110
047400	117	O	79	067400	157	o	111
050000	120	P	80	070000	160	p	112
050400	121	Q	81	070400	161	q	113
051000	122	R	82	071000	162	r	114
051400	123	S	83	071400	163	s	115
052000	124	T	84	072000	164	t	116
052400	125	U	85	072400	165	u	117
053000	126	V	86	073000	166	v	118
053400	127	W	87	073400	167	w	119
054000	130	X	88	074000	170	x	120
054400	131	Y	89	074400	171	y	121
055000	132	Z	90	075000	172	z	122
055400	133	[	91	075400	173	{	123
056000	134	\	92	076000	174	|	124
056400	135	]	93	076400	175	}	125
057000	136	^	94	077000	176	~	126

Table 8–1 (Cont.): ASCII Character Set, Left/Right Byte Equivalents 000-177

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
057400	137	_	95	077400	177	DEL	127

Table 8–2: ASCII Character Set, Left/Right Byte Equivalents 200-377

Left Byte Octal	Right Byte Octal	Right Byte Decimal	Left Byte Octal	Right Byte Octal	Right Byte Decimal
100000	200	128	120000	240	160
100400	201	129	120400	241	161
101000	202	130	121000	242	162
101400	203	131	121400	243	163
102000	204	132	122000	244	164
102400	205	133	122400	245	165
103000	206	134	123000	246	166
103400	207	135	123400	247	167
104000	210	136	124000	250	168
104400	211	137	124400	251	169
105000	212	138	125000	252	170
105400	213	139	125400	253	171
106000	214	140	126000	254	172
106400	215	141	126400	255	173
107000	216	142	127000	256	174
107400	217	143	127400	257	175
110000	220	144	130000	260	176
110400	221	145	130400	261	177

Table 8–2 (Cont.): ASCII Character Set, Left/Right Byte Equivalents 200-377

Left Byte Octal	Right Byte Octal	Right Byte Decimal	Left Byte Octal	Right Byte Octal	Right Byte Decimal
111000	222	146	131000	262	178
111400	223	147	131400	263	179
112000	224	148	132000	264	180
112400	225	149	132400	265	181
113000	226	150	133000	266	182
113400	227	151	133400	267	183
114000	230	152	134000	270	184
114400	231	153	134400	271	185
115000	232	154	135000	272	186
115400	233	155	135400	273	187
116000	234	156	136000	274	188
116400	235	157	136400	275	189
117000	236	158	137000	276	190
117400	237	159	137400	277	191
140000	300	192	160000	340	224
140400	301	193	160400	341	225
141000	302	194	161000	342	226
141400	303	195	161400	343	227
142000	304	196	162000	344	228
142400	305	197	162400	345	229
143000	306	198	163000	346	230
143400	307	199	163400	347	231
144000	310	200	164000	350	232

Table 8–2 (Cont.): ASCII Character Set, Left/Right Byte Equivalents 200-377

Left Byte Octal	Right Byte Octal	Right Byte Decimal	Left Byte Octal	Right Byte Octal	Right Byte Decimal
144400	311	201	164400	351	233
145000	312	202	165000	352	234
145400	313	203	165400	353	235
146000	314	204	166000	354	236
146400	315	205	166400	355	237
147000	316	206	167000	356	238
147400	317	207	167400	357	239
150000	320	208	170000	360	240
150400	321	209	170400	361	241
151000	322	210	171000	362	242
151400	323	211	171400	363	243
152000	324	212	172000	364	244
152400	325	213	172400	365	245
153000	326	214	173000	366	246
153400	327	215	173400	367	247
154000	330	216	174000	370	248
154400	331	217	174400	371	249
155000	332	218	175000	372	250
155400	333	219	175400	373	251
156000	334	220	176000	374	252
156400	335	221	176400	375	253
157000	336	222	177000	376	254
157400	337	223	177400	377	255

Table 8–3: Radix–50 Character Set

Character	First	Second	Third
SPACE	000000	000000	000000
A	003100	000050	000001
B	006200	000120	000002
C	011300	000170	000003
D	014400	000240	000004
E	017500	000310	000005
F	022600	000360	000006
G	025700	000430	000007
H	031000	000500	000010
I	034100	000550	000011
J	037200	000620	000012
K	042300	000670	000013
L	045400	000740	000014
M	050500	001010	000015
N	053600	001060	000016
O	056700	001130	000017
P	062000	001200	000020
Q	065100	001250	000021
R	070200	001320	000022
S	073300	001370	000023
T	076400	001440	000024
U	101500	001510	000025
V	104600	001560	000026
W	107700	001630	000027

Table 8–3 (Cont.): Radix–50 Character Set

Character	First	Second	Third
X	113000	001700	000030
Y	116100	001750	000031
Z	121200	002020	000032
$	124300	002070	000033
.	127400	002140	000034
unused	132500	002210	000035
0	135600	002260	000036
1	140700	002330	000037
2	144000	002400	000040
3	147100	002450	000041
4	152200	002520	000042
5	155300	002570	000043
6	160400	002640	000044
7	163500	002710	000045
8	166600	002760	000046
9	171700	003030	000047

Table 8–4: RT–11 Device Names and Codes

Name	Code	Device
RK	0	RK05 Disk
DT	1	Reserved
EL	2	Error Logger
LP	3	Line Printer
TT,BA	4	Console Terminal or Batch Handler

Table 8–4 (Cont.): RT–11 Device Names and Codes

Name	Code	Device
DL	5	RL01/RL02 Disk
DY	6	RX02 Diskette
PC	7	Reserved
	10	Reserved
MT	11	TM11/TMA11/TU10/TS03 Magtape
RF	12	Reserved
CT	13	Reserved
CR	14	Reserved
	15	Reserved
DS	16	RJS03/Reserved
	17	Reserved
MM	20	TJU16/TU45 Magtape
DP	21	Reserved
DX	22	RX11/RX01 Diskette
DM	23	RK06/RK07 Disk
	24	Reserved
NL	25	Null Device
	26-30	Reserved (DECnet)
	31-33	Reserved (CTS-300)
DD	34	Reserved
MS	35	TS11/TS04 Magtape
PD	36	Reserved
PD	37	Reserved
	40	Reserved

Table 8–4 (Cont.): RT–11 Device Names and Codes

Name	Code	Device
LS	41	Serial Line Printer
MQ	42	Internal Message Handler
DR	43	DRV11J Interface (MRRT)
XT	44	Reserved (MRRT)
	45	Reserved
LD	46	Logical Disk Handler
VM	47	KT11 Pseudo-disk Handler
DU	50	MSCP Disk Class Handler (RA80, RC25)
SL	51	Single-line Editor
DZ	52	RX50 Diskette (Professional 325/350)
DW	53	RD50/RD51 Disk (Professional 350)
PI	54	Professional Interface
SP	55	Transparent Spooler
XC/XL	57	Communication Port (Professional 325/350 or DL(V)-11)
MU	60	TMSCP Magtape Class Handler
NC/NQ/NU	61	Ethernet Class Handler
SD	62	DBG-11 Handler
ST	63	DBG-11 Symbol Table Handler
	64	Reserved
UB	65	UMR Pseudohandler
	66-101	Reserved
AT	102	Address Translation Pseudohandler

Table 8–5: Standard RT–11 File Types

Type	Meaning
.ANS	SYSGEN answer file
.BAC	BASIC intermediate file
.BAD	File with bad blocks
.BAK	Editor backup file
.BAS	BASIC source file
.BAT	BATCH command file
.BLD	SYSGEN build procedure file
.BUP	Backup Utility Program output file
.CND	System generation conditional file
.COM	Indirect command file, IND control file, SIPP indirect file
.CTL	BATCH control file (generated by BATCH from .BAT file)
.CTT	BATCH temporary file
.DAT	Generic data file
.DBL	DIBOL source file
.DEV	SYSGEN handler build procedure
.DIF	SRCCOM, BINCOM output file
.DIR	Directory listing file
.DMP	DUMP output file
.DSK	Logical disk file
.FNT	Font table file
.FOR	FORTRAN source file
.JOB	QUEUE output file
.LDA	Absolute binary file
.LOG	BATCH log file

Table 8–5 (Cont.): Standard RT–11 File Types

Type	Meaning
.LST	Listing file, SLP output file
.MAC	MACRO source file
.MAP	LINK load map
.MLB	MACRO library file
.MON	SYSGEN monitor build procedure
.OBJ	Relocatable binary file
.REL	Foreground job relocatable file
.SAV	Memory image program file
.SLP	SLP command file created by SRCCOM
.SML	System macro library
.SOU	Temporary BATCH file
.STB	Symbol table file
.SYG	Monitor and handler files created by SYSGEN
.SYS	System files and handlers
.TBL	SYSGEN device table file
.TEC	TECO macro file
.TMP	Temporary file
.TXT	Text file

Table 8–6: Interrupt, Trap, and Software Vectors

0,2	Reserved
4,6	CPU errors, odd address trap
10,12	Illegal and reserved instruction trap
14,16	BPT, breakpoint trap

Table 8–6 (Cont.): Interrupt, Trap, and Software Vectors

20,22	IOT, input/output trap
24,26	Power fail
30,32	EMT instructions
34,36	TRAP instructions
40-56	SYSCOM
60,62	DL11: Console terminal input
64,66	DL11: Console terminal output
70,72	Reserved
74,76	Reserved
100,102	KW11-L: Line clock
104,106	KW11-P: Programmable clock
110,112	SP: Spooler handler
114,116	Memory parity errors
120,122	DEQNA
124,126	Reserved
130,132	Reserved
134,136	Reserved
140,142	Reserved
144,146	Reserved
150,152	Reserved
154,156	MSCP
160,162	RL11: RL01/RL02 disk
164,166	Reserved
170,172	Reserved
174,176	Reserved

Table 8–6 (Cont.): Interrupt, Trap, and Software Vectors

200,202	LP11/LS11/LPV11: Line printer
204,206	Reserved
210,212	RK611/RK711: RK06/RK07 disk
214,216	Reserved
220,222	RK11/RKV11: RK05 disk
224,226	Magtape
230,232	Reserved
234,236	Reserved
240,242	PIRQ (programmed interrupt request)
244,246	FPP/FIS floating-point exception
250,252	KT11: Memory management fault
254,256	Reserved
260,262	MU: TMSCP magtape
264,266	RX11/RXV11/RX211/RX2V1: RX01/02 diskette
270,272	Reserved
274,276	Reserved
300,302	Start of floating vector area
320,322	Reserved
324,326	Reserved
330,332	Reserved

Table 8–7: K Equivalents

K-value	Octal	Decimal	K-value	Octal	Decimal
1	2000	1024	51	146000	52224
2	4000	2048	52	150000	53248
3	6000	3072	53	152000	54272
4	10000	4096	54	154000	55296
5	12000	5120	55	156000	56320
6	14000	6144	56	160000	57344
7	16000	7168	57	162000	58368
8	20000	8192	58	164000	59392
9	22000	9216	59	166000	60416
10	24000	10240	60	170000	61440
11	26000	11264	61	172000	62464
12	30000	12288	62	174000	63488
13	32000	13312	63	176000	64512
14	34000	14336	64	200000	65536
15	36000	15360	65	202000	66560
16	40000	16384	66	204000	67584
17	42000	17408	67	206000	68608
18	44000	18432	68	210000	69632
19	46000	19456	69	212000	70656
20	50000	20480	70	214000	71680
21	52000	21504	71	216000	72704
22	54000	22528	72	220000	73728
23	56000	23552	73	222000	74752
24	60000	24576	74	224000	75776

Table 8–7 (Cont.): K Equivalents

K-value	Octal	Decimal	K-value	Octal	Decimal
25	62000	25600	75	226000	76800
26	64000	26624	76	230000	77824
27	66000	27648	77	232000	78848
28	70000	28672	78	234000	79872
29	72000	29696	79	236000	80896
30	74000	30720	80	240000	81920
31	76000	31744	81	242000	82944
32	100000	32768	82	244000	83968
33	102000	33792	83	246000	84992
34	104000	34816	84	250000	86016
35	106000	35840	85	252000	87040
36	110000	36864	86	254000	88064
37	112000	37888	87	256000	89088
38	114000	38912	88	260000	90112
39	116000	39936	89	262000	91136
40	120000	40960	90	264000	92160
41	122000	41984	91	266000	93184
42	124000	43008	92	270000	94208
43	126000	44032	93	272000	95232
44	130000	45056	94	274000	96256
45	132000	46080	95	276000	97280
46	134000	47104	96	300000	98304
47	136000	48128	97	302000	99328
48	140000	49152	98	304000	100352

Table 8–7 (Cont.): K Equivalents

K-value	Octal	Decimal	K-value	Octal	Decimal
49	142000	50176	99	306000	101376
50	144000	51200	100	310000	102400
101	312000	103424	140	430000	143360
102	314000	104448	144	440000	147456
103	316000	105472	148	450000	151552
104	320000	106496	152	460000	155648
105	322000	107520	156	470000	159744
106	324000	108544	160	500000	163840
107	326000	109568	164	510000	167936
108	330000	110592	168	520000	172032
109	332000	111616	172	530000	176128
110	334000	112640	176	540000	180224
111	336000	113664	180	550000	184320
112	340000	114688	184	560000	188416
113	342000	115712	188	570000	192512
114	344000	116736	192	600000	196608
115	346000	117760	196	610000	200704
116	350000	118784	200	620000	204800
117	352000	119808	204	630000	208896
118	354000	120832	208	640000	212992
119	356000	121856	212	650000	217088
120	360000	122880	216	660000	211184
121	362000	123904	220	670000	225280
122	364000	124928	224	700000	229376

Table 8–7 (Cont.): K Equivalents

K-value	Octal	Decimal	K-value	Octal	Decimal
123	366000	125952	228	710000	233472
124	370000	126976	232	720000	237568
125	372000	128000	236	730000	241664
126	374000	129024	240	740000	245760
127	376000	130048	244	750000	249856
128	400000	131072	248	760000	253952
.	.	.	252	770000	258048
.	.	.	256	1000000	262144
132	410000	135168			
136	420000	139264			