```
.TITLE  LOCK.MAC
;THIS EXAMPLE SHOWS THE USAGE OF .LOCK, .UNLOCK, AND THEIR
;INTERACTION WITH THE SYSTEM.
        .MCALL  .LOCK,.UNLOCK,.LOOKUP
        .MCALL  .SETTOP,.PRINT,.EXIT

START:

SYSPTR=54

        .SETTOP #^O54           ;TRY FOR ALL OF MEMORY
        MOV     R0,TOP          ;R0 HAS THE TOP
        .LOCK                   ;BRING USR INTO MEMORY
        .LOOKUP #LIST,#0,#FILE1 ;LOOKUP A FILE ON CHANNEL 0
        BCC     1$              ;ON ERROR, PRINT A
2$:     .PRINT  #LMSG           ;MESSAGE AND EXIT
        .EXIT
1$:     MOV     #LIST,R0
        INC     (R0)            ;DO LOOKUP ON CHANNEL 1
        MOV     #FILE2,2(R0)    ;NEW POINTER
        .LOOKUP         ;ALL ARGS ARE FILLED IN
        BCS     2$
        .UNLOCK                 ;NOW RELEASE USR
        .EXIT


LIST:   .BLKW   3               ;SPACE FOR ARGUMENTS
FILE1:  .RAD50  /DK /
        .RAD50  /FILE1 MAC/
FILE2:  .RAD50  /DK /
        .RAD50  /FILE2 MAC/
TOP:    .WORD   0
LMSG:   .ASCIZ  /LOOKUP ERROR/
        .EVEN

        .END    START
```


2.4.26 .LOOKUP

The .LOOKUP request associates a specified channel with a device and existing file, for the purpose of performing I/O operations. The channel used is then busy until one of the following requests is executed:

```
.CLOSE
.SAVESTATUS
.SRESET
.HRESET
.PURGE
.CSIGEN    (if the channel is in the range 0-10 octal)
```

Note that if the program is overlaid, channel 15 (decimal) or 17 (octal) is used by the overlay handler and should not be modified.

If the first word of the file name in dblk is 0 and the device is a file-structured device, absolute block 0 of the device is designated as the beginning of the file. This technique is called a non-file-structured .LOOKUP, and allows I/O operations to access any physical block on the device. If a file name is specified for a device that is not file-structured (such as PC:FILE.TYP), the name is ignored.

The handler for the selected device must be in memory for a .LOOKUP. 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, R0 contains 0 for the length.

NOTE

Care should be exercised when doing a non-file-structured .LOOKUP on a file-structured device, since corruption of the device directory can occur and effectively destroy the disk. (The RT-11 directory starts in absolute block 6.)

In particular, avoid doing a .LOOKUP or .ENTER with a file specification that is missing the file value. If the device type is not known in advance and is to be entered from the keyboard, include a dummy file name with the .LOOKUP or .ENTER, even when it is assumed that the device is always non-file-structured.

Macro Call: .LOOKUP area,chan,dblk,seqnum

where:

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

chan — a channel number in the range 0-377(octal).

dblk — the address of a four-word Radix-50 descriptor of the file to be operated upon.

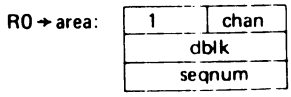
seqnum — file number. For cassette operations, if this argument is blank, a value of 0 is assumed. For magtape, it describes a file sequence number that can have the following values.

- -1 means suppress rewind and search for a file name from the current tape position. If the position is not known, then the handler executes a positioning algorithm that involves backspacing until an end-of-file label is found.*
- 0 means rewind the magtape and space forward until the file name is found.

* It is important that -1 be specified and no other negative number.

n where n is any positive number. This means position the tape at file sequence number n. If the file represented by the file sequence number is greater than two files away from the beginning-of-tape, then a rewind is performed. If not, the tape is backspaced to the beginning of the file.

Request Format:



Errors:

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

Example:

```
.TITLE  LOOKUP.MAC
;IN THIS EXAMPLE, THE FILE "DATA.001" ON DEVICE DT3:
;IS OPENED FOR INPUT  ON CHANNEL 7.
        .MCALL  .FETCH,.LOOKUP,.PRINT,.EXIT

START:
ERRBYT=52
        .FETCH  #HSPACE,#DT3N   ;GET DEVICE HANDLER
        BCS     FERR            ;DT3 IS NOT AVAILABLE
        .LOOKUP #LIST,#7,#DT3N  ;LOOKUP THE FILE
                                ;ON CHANNEL 7
        BCC     LDONE           ;FILE WAS FOUND
        TSTB    @#ERRBYT                ;ERROR, WHAT'S WRONG?
        BNE     NFD             ;FILE NOT FOUND
        .PRINT  #CAMSG          ;PRINT  'CHANNEL ACTIVE'
        .EXIT
NFD:    .PRINT  #NFMSG          ;FILE NOT FOUND
        .EXIT
CAMSG:  .ASCIZ  /CHANNEL ACTIVE/
NFMSG:  .ASCIZ  /FILE NOT FOUND/ ;ERROR MESSAGES
DTMSG:  .ASCIZ  /DT3 NOT AVAILABLE/
        .EVEN
FERR:   .PRINT  #DTMSG
        .EXIT
LDONE:                          ;PROGRAM CAN NOW
                                ;ISSUE READS  AND
                                ;WRITES TO FILE
                                ;DATA.001 VIA
                                ;CHANNEL 7

        .EXIT

LIST:   .BLKW   5
DT3N:   .RAD50  "DT3"           ;DEVICE
        .RAD50  "DAT"           ;FILENAME
        .RAD50  "A  "           ;FILENAME
        .RAD50  "001"           ;EXTENSION

HSPACE:                 ;RESERVED SPACE FOR DT
        .=.+400                 ;HANDLER

        .END    START
```

.MFPS/.MTPS

2.4.27 .MFPS/.MTPS

The .MFPS and .MTPS macro calls allow processor-independent user access to the processor status word. The contents of R0 are preserved across either call.

The .MFPS call is used to read 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.

In the XM monitor, .MFPS and .MTPS can be used only by privileged jobs; they are not available for use by virtual jobs.

Macro Call: .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.

The .MTPS call is used to set the priority, condition codes, and trace trap bit with the value designated in the call.

Macro Call: .MTPS addr

where: addr — is the address of the word to be placed in the processor status word; if addr 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 addr is present. If addr is not present, the user should set the stack to the appropriate value. In either case, the lower byte on the stack is put in the processor status word.

Notes:

It is possible to do .MFPS and .MTPS operations and also access the condition codes by a special technique. The routines $MFPS and $MTPS are accessed by monitor fixed offsets, and condition codes are normally destroyed in the macros when the addresses of the routines are constructed. If the programmer constructs them in advance, calls can be made that return condition codes. An example of this technique follows:

```
                                        ;Fixed offset values
                                        ;$MTPS 360
                                        ;$MFPS 362
            ADD @#54,PMTPS              ;Relocate PMTPS
            ADD @#54,PMFPS              ;Relocate PMFPS
                .
                .
                .
            JSR PC,@PMFPS               ;Put PSW
                .
                .
                .
            JSR PC,@PMTPS               ;Put PSW
                .
                .
                .
PMTPS:      .WORD 360
PMFPS:      .WORD 362
                .
                .
                .
```

Errors:

None.

Example:

```
        .TITLE  MFPS
        .MCALL  .MFPS,.MTPS,.EXIT

START:  JSR     PC,PICKQ        ;PICK A QUEUE ELEMENT
                .
                .
                .
        .EXIT

PICKQ:  .MFPS                   ;SAVE PREVIOUS PRIORITY ON STACK
        MOV     #QHEAD,R4       ;POINT TO QUEUE HEAD
        .MTPS   #340            ;RAISE PRIORITY TO 7
        MOV     @R4,R5          ;R5 POINTS TO NEXT ELEMENT
        BEQ     10$             ;NO MORE ELEMENTS AVAILABLE
        MOV     @R5,@R4         ;RELINK THE QUEUE
        .MTPS                   ;RESTORE PREVIOUS PRIORITY
        CLZ                     ;FLAG SUCCESS
        BR      20$
10$:    .MTPS                   ;RESTORE PREVIOUS PRIORITY
        SEZ                     ;INDICATE FAILURE
20$:    RTS     PC

QHEAD:  .WORD   Q1              ;QUEUE HEAD

;THREE QUEUE ELEMENTS
Q1:     .WORD   Q2,0,0
Q2:     .WORD   Q3,0,0
Q3:     .WORD   0,0,0

        .END    START
```

.MRKT

2.4.28 .MRKT (FB and XM Only; SJ Monitor SYSGEN Option)

The .MRKT request schedules a completion routine to be entered after a specified time interval (clock ticks) has elapsed. .MRKT is an optional feature in the SJ monitor; timer support must be selected at system generation time to obtain it.

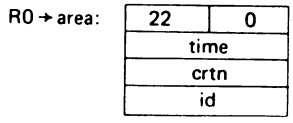
.MRKT requests require a queue element taken from the same list as the I/O queue elements. The element is in use until either the completion routine is entered or a cancel mark time request is issued. The user should allocate enough queue elements to handle at least as many mark time and I/O requests as he expects to have pending simultaneously.

Macro Call: `.MRKT area,time,crtn,id`

where:

- **area** is the address of a four-word EMT argument block.
- **time** is the pointer to the two words containing the time interval (high-order first, low-order second).
- **crtn** is the entry point of a completion routine.
- **id** is a number assigned by the user to identify the particular request to the completion routine and to any cancel mark time requests. The number must not be within the range 177400 - 177777; these are reserved for system use. The number need not be unique (several .MRKT requests can specify the same id). On entry to the completion routine, the id number is in R0.

Request Format:



Errors:

Code	Explanation
0	No queue element was available.

Example:

```
.TITLE  MRKT.MAC
;IN THIS EXAMPLE, A MARK TIME IS SET UP TO TIME OUT AN I/O
;TRANSFER. IF THE MARK TIME EXPIRES BEFORE THE TRANSFER IS DONE
;A MESSAGE IS PRINTED. IF THE I/O TRANSFER COMPLETES BEFORE THE
;MARK TIME, THE MARK TIME IS CANCELLED.
        .MCALL  .READ,.WAIT,.MRKT,.CMKT
        .MCALL  .QSET,.PRINT,.EXIT,.LOOKUP

ST:     .LOOKUP #AREA,#0,#FILE  ;OPEN A FILE
        BCS     LKERR           ;FILE NOT FOUND
        MOV     #AREA,-(SP)     ;EMT LIST TO STACK
        .QSET   #QUEUE,#5       ;ALLOCATE 5 MORE ELEMENTS
        .MRKT   (SP),#INTRVL,#MRTN,#1 ;SET TIMER GOING
        BCS     NOMRKT          ;FAILED.
        .READ   #RDLST          ;START  I/O TRANSFER
        BCS     RDERR
        .WAIT   #0              ;AND WAIT A WHILE.
        .CMKT   (SP),#1         ;SEE IF MARK TIME IS
                                ;DONE.
        BCS     NOTDUN          ;FAILED. THAT MEANS THAT
                                ;THE MARK TIME ALREADY
                                ;EXPIRED.

        .EXIT


MRTN:   .CMKT   (SP),#1         ;OK, KILL THE TIMER.
        .PRINT  #FAIL           ;DON'T WORRY ABOUT AN
                                ;ERROR HERE.
        RTS     PC
LKERR:  .PRINT  #LM
        .EXIT
RDERR:  .PRINT  #RDMSG
        .EXIT
NOTDUN: .PRINT  #FAIL
        .EXIT
NOMRKT: .PRINT  #NOQ
        .EXIT
NOQ:    .ASCIZ  /NO QUEUE ELEMENTS AVAILABLE/
FAIL:   .ASCIZ  /MARK TIME COMPLETED BEFORE TRANSFER/
LM:     .ASCIZ  /LOOKUP ERROR/
RDMSG:  .ASCIZ  /READ ERROR/
        .EVEN
INTRVL: .WORD   0,13.           ;ALLOW 13 CLOCK
                                ;TICKS FOR TRANSFER.
QUEUE:  .BLKW   5*7             ;AREA FOR QUEUE ELEMENTS
AREA:   .BLKW   5               ;A FEW WORDS FOR EMT LIST
FILE:   .RAD50  /DK FILE  TST/
RDLST:  .BYTE   0               ;CHANNEL 0
        .BYTE   10              ;A READ
BLOCK:  .WORD   0               ;BLOCK #
        .WORD   BUFF            ;BUFFER
        .WORD   256.            ;1 BLOCK
        .WORD   1
BUFF:   .BLKW   256.

        .END    ST
```

.MTATCH

2.4.29 .MTATCH (FB and XM Monitor SYSGEN Option)

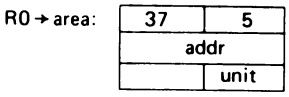
This request attaches a terminal for exclusive use by the requesting job. This operation must be performed before any job can use a terminal with multi-terminal programmed requests.

Macro Call: .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 SYSGEN option.)
unit	is the logical unit number (lun) of the terminal.

Request Format:



Errors:

When the carry bit is set, the following errors are returned in byte 52:

Codes	Explanation
2	Non-existent lun
3	Illegal request, function code out of range
4	Unit attached by another job
5	In the XM monitor, the optional status word address is not in valid user virtual address space.

Example:

The following example demonstrates use of the multi-terminal asynchronous status option by polling all terminals to determine which terminals are on-line. The example in the section on the .MTSET request also illustrates the .MTATCH request.

```
        .MCALL  .MTATCH,.MTPRNT,.EXIT

AS.CAR  = 200                   ;IF SET, INDICATES CARRIER
                                ; PRESENT.
START:
        CLR     R1              ;INITIALIZE LUN
        MOV     #AST,R2         ;R2 -> ASYNCHRONOUS STATUS WORD
10$:    .MTATCH #MTA,R2,R1      ;ATTEMPT TO ATTACH TERMINAL
        BCC     15$             ;BR IF SUCCESSFUL
        CLRB    TAI(R1)         ;SIGNAL ATTACH UNSUCCESSFUL
        BR      20$             ;GO TRY NEXT LUN
15$:    MOVB    #1,TAI(R1)      ;SIGNAL ATTACH SUCCESSFUL
        BIT     #AS.CAR,(R2)    ;TERMINAL ON LINE?
        BEQ     20$             ;BR IF HUNG UP
        .MTPRNT #MTA,#HELLO,R1  ;SAY HELLO
20$:    ADD     #2,R2           ;R2 -> NEXT AST WORD
        INC     R1              ;NEXT LUN
        CMP     R1,#16.         ;DONE?
        BLO     10$             ;BR IF NOT
        .EXIT
AST:    .BLKW   16.             ;ASYNCHRONOUS TERMINAL STATUS WORDS
MTA:    .BLKW   3               ;PARAMETER BLOCK FOR EMT'S
HELLO:  .ASCIZ  /WE'RE ON THE AIR/
TAI:    .BLKB   16.             ;0 => TERMINAL NOT ATTACHED
                                ;1 => TERMINAL ATTACHED
        .END    START
```

.MTDTCH

2.4.30 .MTDTCH (FB and XM Monitor SYSGEN Option)

This request detaches a terminal from one job and makes it available for other jobs. When a terminal is detached, it is deactivated and unsolicited interrupts are ignored. Input is disabled immediately, but any characters in the output buffer are allowed to print. Attempts to detach a terminal attached by another job results in an error. However, attempts to detach an unattached terminal are ignored.

Macro Call: .MTDTCH area,unit

where: area — is a three-word EMT argument block.

unit — is the logical unit number (lun) of the terminal.

Request Format:

R0→area:

37	6
(unused)	
--	unit

Errors:

When the carry bit is set, the following errors are returned in byte 52.

Code	Explanation
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range

Example:

```
.TITLE  MTDTCH.MAC
        .MCALL  .MTDTCH,.MTPRNT,.MTATCH,.EXIT,.PRINT

START:
        .MTATCH #MTA,#0,#3      ;ATTACH TO LUN 3
        BCC     1$              ;ATTACH ERROR
        .MTPRNT #MTA,#MESS,#3   ;PRINT MESSAGE
        .MTDTCH #MTA,#3         ;DETACH LUN 3
        .EXIT
1$:     .PRINT  #ATTERR         ;ATTACH ERROR
                                ;(PRINTED ON CONSOLE)
        .EXIT
ATTERR: .ASCIZ/ATTACH ERROR/
MESS:   .ASCIZ/DETACHING TERMINAL/
        .EVEN
MTA:    .BLKW   3
        .END    START
```

.MTGET

2.4.31 .MTGET (FB and XM Monitor SYSGEN Option)

This request returns the status of the specified terminal unit to the caller.

When the program returns from the request, the status block contains the following information:

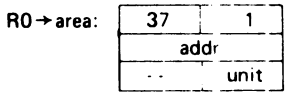
Byte Offset	Contents
0	Terminal configuration word 1. The bit definitions are the same as those for the .MTSET request.
2	Terminal configuration word 2
4	Character requiring fillers
5	Number of fillers
6	Carriage width (byte).
7	Current carriage position (byte).

Macro Call: .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 (see Section 2.4.36).
- unit — is the logical unit number (lun) of the terminal whose status is requested.

Request Format:



Errors:

When the carry bit is set, the following errors are returned in byte 52.

Code	Explanation
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range.
5	In the XM monitor, the optional status word address is not in valid user virtual address space.

Example:

See the example at the end of the .MTSET section.

2.4.32 .MTIN (FB and XM Monitor SYSGEN Option)

The .MTIN request is the multi-terminal form of the .TTYIN request. It does not use a queue element, but it does require an argument block. The .MTIN request moves one or more characters from the input ring buffer to the user's buffer specified by addr. The terminal must be attached and an updated user buffer address is returned in R0 if the request is successful. If bit 6 is set in the M.TSTS (see MTSET) word, the .MTIN request returns immediately with the carry bit set (code 0) if there is no input available (no line if bit 12 is clear; no characters in buffer if bit 12 is set in M.TSTS). If these conditions do not exist, the .MTIN request waits until input is available and the job is suspended until input is available.

If a multiple character request was made and the number of characters requested is not available, the request can either wait for the characters to become available, or it can return with a partial transfer. If bit 6 of M.TSTS is clear, the request waits for more characters. If bit 6 is set, the request returns with a partial transfer. In the latter case, R0 contains the updated buffer address (pointing past the last character transferred), the C bit is set, and the error code is 0.

The .MTIN request has the following form:

Macro Call: .MTIN area,addr,unit,chrcnt

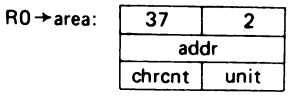
where: area is the address of a three-word EMT argument block.

addr is the byte address of the user buffer.

unit — is the logical unit number of the terminal input.

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

Request Format:



Errors:

When the carry bit is set, the following errors are returned in byte 52.

Code	Explanation
0	No input available -- bit 6 is set in JSW (for system console) or M.TSTS
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range
5	In the XM monitor, the optional status word address is not in valid user virtual address space.

Example:

See the example at the end of the .MTSET section.

.MTOUT

2.4.33 .MTOUT (FB and XM Monitor SYSGEN Option)

This request is the complement to the .MTIN request. It is the multi-terminal form of the .TTYOUT request. It does not use a queue element, but does require an argument block. The .MTOUT request moves one or more characters from the user's buffer to the output ring buffer of the terminal. The terminal must be attached. An updated user buffer address is returned in R0 if the request is successful. If there is no room in the output ring buffer, the carry bit is set and an error code of 0 is returned in byte 52 if bit 6 is set in M.TSTS. Otherwise, the request waits for room and the job is suspended until room becomes available.

If a multiple character request was made and there is not enough room in the output ring buffer to transfer the requested number of characters, the request can either wait for enough room to become available, or it can return with a partial transfer. If bit 6 in M.TSTS is clear, the request waits until it can complete the full transfer. If bit 6 is set, the request returns with a partial transfer. In the latter case, R0 contains the updated buffer address (pointing past the last character transferred), the C bit is set, and the error code is 0.

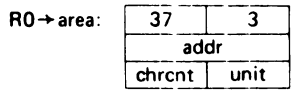
The .MTOUT request has the following form:

Macro Call: .MTOUT area,addr,unit,chrcnt

where:

area	is the address of a three-word EMT argument block.
addr	is the address of the caller's input buffer.
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).

Request Format:



Errors:

Code	Explanation
0	No room in output buffer
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range

Example:

See the example at the end of the .MTSET section.

.MTPRNT

2.4.34 .MTPRNT (FB and XM Monitor SYSGEN Option)

This request operates in a multi-terminal environment in the same way as the .PRINT request. It allows one or more lines to be printed at the specified terminal (see Section 2.4.36 for more details). The request does not return until the transfer is complete.

Macro Call: .MTPRNT area,addr,unit

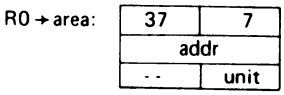
where:

area	is the address of a three-word EMT argument block.
addr	is the starting address of the character string to be printed. The string must be terminated with a null byte or a 200 byte, similar to the string used with the .PRINT request. The null byte causes a carriage return/line feed combination to be printed after the string. The 200 byte suppresses the carriage return/line feed combination and leaves the carriage positioned after the last character of the string.

For example: .ASCII /string/<200>
or .ASCIZ /string/

unit — is the unit number associated with the terminal.

Request Format:



Errors:

When the carry bit is set, the following errors are returned in byte 52.

Code	Explanation
1	Illegal unit number, lun not attached
2	Non-existent lun
5	In the XM monitor, the optional status word address is not in valid user virtual address space.

Example:

See the example at the end of the .MTSET section.

.MTRCTO

2.4.35 .MTRCTO (FB and XM Monitor SYSGEN Option)

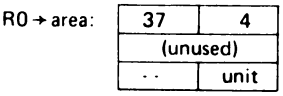
The .MTRCTO request operates in a multi-terminal environment in the same way as the .RCTRLO request. It resets the CTRL/O switch of the specified terminal and enables terminal output.

Macro Call: .MTRCTO area,unit

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

unit — is the unit number associated with the terminal.

Request Format:



Errors:

When the carry bit is set, the following errors are returned in byte 52.

Code	Explanation
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range

Example:

See the example at the end of the .MTSET section.

.MTSET

2.4.36 .MTSET (FB and XM Monitor SYSGEN Option)

This multi-terminal request allows the user program to set terminal and line characteristics. It also determines the input/output mode of the terminal service requests for the specified terminal. This request has the following form:

Macro Call: .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 requested.
unit	is the logical unit number associated with the line and terminal.

The user is required to supply information to the status block. It has the following structure:

R0→area:

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

Offset	Description
0 (M.TSTS)	Terminal configuration word 1
2 (M.TST2)	Reserved for future use
4 (M.TFIL)	Character requiring fillers
5 (M.FCNT)	Number of fillers
6 (M.TWID)	Carriage width
7 (M.TSTW)	Terminal state byte

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

Mask	Bit	Meaning
1	0	Hardware tab
2	1	Output RET/LF when carriage width exceeded
4	2	Hardware form feed
10	3	Process CTRL/F and CTRL/B as normal characters
20	4	Enable escape sequence processing
40	5	Filter escape sequences
100	6	Inhibit TT wait (similar to TCBIT$ in the JSW)
200	7	XON/XOFF processing enabled
7400	8-11	Line speed (baud rate) mask (defined in full below)
10000	12	Character mode input (similar to TTSPC$ in the JSW)
20000	13	Terminal is remote (Read Only bit)
40000	14	Lower to upper case conversion disabled
100000	15	Use backspace for rubout (video type display)

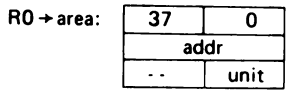
Bits 8 through 11 of configuration word 1 (M.TSTS) indicate the terminal baud rate (DZ11 only). The values are as follows:

Octal Value of Line Speed Mask (M.TSTS bits 11-8)	Baud Rate
0000	50
0400	75
1000	110
1400	134.5
2000	150
2400	300
3000	600
3400	1200
4000	1800
4400	2000
5000	2400
5400	3600
6000	4800
6400	7200
7000	9600
7400	(unused)

The bit definitions for M.TSTW are as follows:

Value	Bit	Meaning
2000	10	Terminal is shared console
4000	11	Terminal has hung up
10000	12	Terminal interface is DZ11
40000	14	Double CTRL/C was struck
100000	15	Terminal is acting as console (local DL11 only)

Request Format:



Errors:

Code	Explanation
1	Illegal unit number, lun not attached
2	Non-existent lun
3	Illegal request, function code out of range
5	In the XM monitor, the optional status word address is not in valid user virtual address space

Example:

This program checks and changes the characteristics of a particular terminal.

```
.MCALL  .MTATCH,.MTPRNT,.MTGET,.MTIN,.MTOUT,.MTSET,.EXIT
.MCALL  .PRINT,.MTRCTO
HNGUP$  = 4000                  ;INDICATES TERMINAL IS HUNG UP
TTSPC$  = 10000                 ;SPECIAL MODE
TTLC$   = 40000                 ;LOWER CASE MODE
AS.INP  = 40000                 ;INDICATES INPUT AVAILABLE
M.TSTS  = 0                     ;TERMINAL STATUS WORD
M.TSTW  = 7                     ;TERMINAL STATE BYTE

START:
        CLR     R1              ;INITIALIZE LUN
        MOV     #AST,R2         ;R2 -> ASYNCHRONOUS TERMINAL STATUS WORDS
10$:    .MTATCH #MTA,R2,R1      ;ATTACH TERMINAL
        BCC     20$             ;BR IF SUCCESSFUL
        CLRB    TAI(R1)         ;SIGNAL ATTACH FAILED, DON'T CARE WHY
        BR      30$             ;PROCEED WITH NEXT LUN
20$:    MOVB    #1,TAI(R1)      ;ATTACH SUCCESSFUL
        MOV     R1,R3           ;COPY LUN
        ASL     R3              ;MULTIPLY BY 3 TO PRODUCE OFFSET
        ASL     R3              ;TO THE TERMINAL STATUS BLOCK
        ASL     R3              ;
        ADD     #TSB,R3         ;R3 -> LUN'S TSB BLOCK
        .MTGET  #MTA,R3,R1      ;GET LUN'S STATUS
        BIS     #TTSPC$+TTLC$,M.TSTS(R3)        ;SET SPECIAL MODE & LOWER CASE
        .MTSET  #MTA,R3,R1      ;SET LUN'S STATUS
        BITB    #HNGUP$/400,M.TSTW(R3)  ;TERMINAL ON LINE?
        BNE     30$             ;BR IF HUNGUP
        .MTRCTO #MTA,R1         ;RESET CTRL/O
        .MTPRNT #MTA,#HELLO,R1  ;SAY HELLO
30$:    ADD     #2,R2           ;R2 -> NEXT AST WORD
        INC     R1              ;NEXT LUN
        CMP     R1,#16.         ;DONE?
        BLO     10$             ;BR IF NOT

LOOP:                           ;READ & ECHO FOREVER UNLESS ERROR OCCURS
        CLR     R1              ;INITIALIZE LUN
        MOV     #AST,R2         ;R2 -> AST WORDS
10$:    TSTB    TAI(R1)         ;TERMINAL ATTACHED?
        BEQ     20$             ;BR IF NOT
        BIT     #AS.INP,(R2)    ;ANY INPUT?
        BEQ     20$             ;BR IF NOT
        .MTIN   #MTA,#MTCHAR,R1,#1 ;READ A CHAR
        BCS     ERR             ;BR IF SOME ERROR
        .MTOUT  #MTA,#MTCHAR,R1,#1  ;OUTPUT CHAR
        BCS     ERR             ;BR IF SOME ERROR
20$:    ADD     #2,R2           ;POINT TO NEXT AST WORD
        INC     R1              ;NEXT LUN
        CMP     R1,#16.         ;DONE ALL
        BLO     10$             ;BR IF NOT
        BR      LOOP            ;REPEAT FOREVER
ERR:    .PRINT  #UNEXP          ;UNEXPECTED ERROR
        .EXIT
AST:    .BLKW   16.             ;ASYNCHRONOUS TERMINAL STATUS WORDS
                                ;1 PER LUN
TAI:    .BLKB   16.             ;TERMINAL ATTACHED INDICATOR LIST
                                ;1 BYTE PER LUN. 0=> NOT ATTACHED
        .EVEN
MTA:    .BLKW   4               ;EMT AREA
MTCHAR: .BYTE   0               ;INPUT STORAGE FOR CHARACTER READ
HELLO:  .ASCIZ  /HAVE A GOOD DAY/
UNEXP:  .ASCIZ  /UNEXPECTED ERROR, PROGRAM ABORTING/
        .EVEN
TSB:    .BLKW   16.*4.          ;TERMINAL STATUS BLOCKS 16. BLOCKS OF 4 WORDS
        .END    START
```

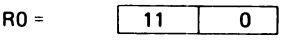
.MWAIT

2.4.37 .MWAIT (FB and XM Only)

This request is similar to the .WAIT request. .MWAIT, however, suspends execution until all messages sent by the other job have been received. It provides a means for ensuring that a required message has been processed. It should be used primarily in conjunction with the .RCVD or .SDAT modes of message handling, where no action is taken when a message is completed.

Macro Call: .MWAIT

Request Format:



Errors:

None.

Example:

```
.TITLE  MWAIT.MAC
;THIS PROGRAM REQUESTS A MESSAGE, DOES SOME INTERMEDIATE PROCESSING,
;AND THE WAITS UNTIL THE MESSAGE IS ACTUALLY SENT.
        .MCALL  .MWAIT,.RCVD,.EXIT,.PRINT

#WORDS=255.
START:
        .RCVD   #AREA,#RBUFF,#WORDS ;GET MESSAGE.

                                    ;INTERMEDIATE PROCESS

        MOV     #RBUFF+2,R5
        .MWAIT                      ;MAKE SURE WE HAVE IT.
        CMPB    (R5)+,#'A           ;FIRST CHARACTER AN A?
        BNE     BADMSG              ;NO, INVALID MESSAGE

        .EXIT
BADMSG: .PRINT  #MSG
        .EXIT
MSG:    .ASCIZ  /BAD MESSAGE/
AREA:   .BLKW   10
RBUFF:  .BLKW   256.
        .EVEN
        .END    START
```

.PRINT

2.4.38 .PRINT

The .PRINT request causes output to be printed at the console terminal. When a foreground job is running and a change occurs in the job producing output, a B> or F> appears. Any text following the message has been printed by the job indicated (foreground or background) until another B> or F> is 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 carriage return/line feed combination. If a 200 byte terminates the string, no carriage return/line feed combination is generated.

Control returns to the user program after all characters have been placed in the output buffer.

The foreground job issues a message immediately using .PRINT no matter what the state of the background job. Thus, for urgent messages, .PRINT should be used (rather than .TTYIN or .TTOUTR). The .PRINT request forces a console switch and guarantees printing of the input line. If a background job is doing a prompt and has printed an asterisk but no carriage return/line feed combination, the console belongs to the background and .TTYOUTs from the foreground are not printed until a carriage return is typed to the background. The foreground job can force its message through by doing a .PRINT instead of the .TTYOUT.

Macro Call: .PRINT addr

where: addr is the address of the string to be printed.

Errors:

None.

Example:

```
.TITLE  PRINT.MAC
        .MCALL  .PRINT,.EXIT

START:
        .PRINT  #S2
        .PRINT  #S1

        .EXIT

S1:     .ASCIZ  /THIS WILL HAVE CR-LF FOLLOWING/
S2:     .ASCII  /THIS WILL NOT HAVE CR-LF/
        .BYTE   200
        .EVEN

        .END    START
```

.PROTECT/.UNPROTECT

2.4.39 .PROTECT/.UNPROTECT (FB and XM Only)

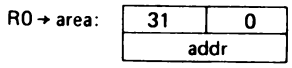
The .PROTECT request is used by a job to obtain exclusive control of a vector (two words) in the region 0-476. If it is successful, it indicates that the locations are not currently in use by another job or by the monitor, in which case the job can place an interrupt address and priority into the protected locations and begin using the associated device.

Macro Call: .PROTECT area,addr

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

addr — is the address of the word pair to be protected. The argument, addr, must be a multiple of four, and must be less than 476 (octal). The two words at addr and addr+2 are protected.

Request Format:



Errors:

Code	Explanation
0	Protect failure; locations already in use.
1	Address greater than 476 or not a multiple of 4.

Example:

```
.TITLE  PROTEC.MAC
;THIS EXAMPLE SHOWS THE USE OF .PROTECT TO GAIN CONTROL
;OF THE UDC11 VECTORS.
        .MCALL  .PROTECT,.PRINT,.EXIT
ST:     MOV     #AREA,@(SP)
        MOV     #234,R5         ;UDC VECTOR ADDRESS
        .PROTECT (SP),R5        ;PROTECT 234,236
        BCS     ERR             ;YOU CAN'T
        MOV     #UDCINT,(R5)+   ;INITIALIZE THE VECTORS.
        MOV     #340,(R5)       ;AT LEVEL 7

        .EXIT
ERR:    .PRINT  #NOVEC
        .EXIT
AREA:   .BLKW   5
NOVEC:  .ASCIZ  /VECTORS ALREADY IN USE/
        .EVEN

UDCINT: RTT
        .END    ST
```

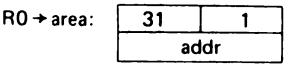
The .UNPROTECT request is the complement of the .PROTECT request. It cancels any protected vectors in the 0 to 476 area.

Macro Call: .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 cancelled.

Request Format:



Errors:

Code	Explanation
1	Address (addr) is greater than 476(octal) or is not a multiple of four.

Example:

```
.TITLE  UNPRO.MAC
;.UNPROT ALLOWS ANOTHER JOB TO USE THE
;SAME VECTORS THAT WERE PREVIOUSLY
;PROTECTED WITH A .PROTECT REQUEST

        .MCALL  .PROTECT,.UNPROTECT,.EXIT,.PRINT
START:
        .PROTECT        #AREA,#234      ;PROTECT 234 & 236
        BCS     INUSE                   ;PROBABLY IN USE
                                        ;BY ANOTHER JOB

;PROGRAM NOW PROCEEDS TO USE THE VECTORS
;WHEN DONE, A .UNPROTECT IS ISSUED, FREEING
;THE VECTORS  FOR USE BY ANOTHER JOB

        .UNPROTECT      #AREA,#234      ;UNPROTECT 234 & 236
        .EXIT
INUSE:  .PRINT  #ERR
        .EXIT
AREA:.BLKW      2
ERR:    .ASCIZ/PROTECT ERROR/
        .EVEN
        .END    START
```

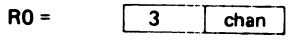
.PURGE

2.4.40 .PURGE

The .PURGE request is used to deactivate a channel without performing a .HRESET, .SRESET, .SAVESTATUS, or .CLOSE request. It merely frees a channel without taking any other action. If a tentative file has been .ENTERed on the channel, it is discarded. Purging an inactive channel acts as a no-op.

Macro Call: .PURGE chan

Request Format:



Errors:

None.

Example:

```
.TITLE  PURGE.MAC
;THIS EXAMPLE VERIFIES THAT CHANNEL 0-7 ARE FREE.
        .MCALL  .PURGE,.EXIT

START:
        CLR     R1              ;START WITH CHANNEL 0
1$:     .PURGE  R1              ;PURGE A CHANNEL
        INC     R1              ;BUMP TO NEXT CHANNEL
        CMP     R1,#8.          ;IS IT AT CHANNEL 8 YET?
        BLO     1$              ;NO, KEEP GOING

        .EXIT
        .END    START
```

.QSET

2.4.41 .QSET

All RT-11 I/O transfers are done through a centralized queue management system. Each non-synchronous transfer request such as a .WRITE requires a queue element until it completes. If I/O traffic is very heavy and not enough queue elements are available, the program issuing the I/O requests can be blocked until a queue element becomes available. In an FB system, the other job can run while the first job waits for the element.

The .QSET request is used to make the RT-11 I/O queue larger (that is, to add available entries to the queue). A general rule to follow is that each program should contain one more queue element than the total number of I/O requests that will be active simultaneously on different channels. Timing and message requests such as .TWAIT and .SDAT also cause elements to be used and must be considered when allocating queue elements for a program. Note that if synchronous I/O is done (.READW/.WRITW, etc.) and no timing requests are done, no additional queue elements need be allocated.

Each time .QSET is called, a contiguous area of memory is divided into seven-word segments (10-word segments for the XM monitor) and is added to the queue for that job. .QSET can be called as many times as required. The queue set up by multiple .QSET requests is a linked list. Thus, .QSET need not be called with strictly contiguous arguments. The space used for the new elements is allocated from the user's program space. Care must be taken so that the program in no way alters the elements once they are set up. The .SRESET and .HRESET requests discard all user-defined queue elements; therefore any .QSETs must be reissued.

Care should also be taken to allocate sufficient memory for the number of queue elements requested. The elements in the queue are altered asynchronously by the monitor; if enough space is not allocated, destructive references occur in an unexpected area of memory. Other restrictions on the placement of queue elements are that the USR must not swap over them and they must not be in an overlay region. For jobs that run under the XM monitor, queue elements must be allocated in the lower 28K words of memory, since they must be accessible in kernel mapping.

NOTE

Programs that are to run in both FB and XM environments should always allocate 10 words for each queue element.

The following requests require queue elements:

.TWAIT	.WRITE
.MRKT	.WRITC
.READ	.WRITW
.READC	.SDAT
.READW	.SDATC
.RCVD	.SDATW
.RCVDC	
.RCVDW	

Macro Call: .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 FB monitor, each queue entry is seven words long; hence the space set aside for the queue should be len*7 words. In the XM monitor, 10 words per queue element are required.

Errors:

None.

Example:

```
.TITLE  QSET.MAC
        .MCALL  .QSET,.EXIT

START:
        .QSET   #Q1,#5          ;ADD 5 ELEMENTS TO THE QUEUE
                                ;STARTING AT Q1
        .QSET   #Q3,#3          ;AND 3 MORE AT Q3.

        .EXIT

Q1:     .BLKW   7*5.            ;FIRST QUEUE AREA (35 DECIMAL WORDS)
Q3:     .BLKW   7*3.            ;SECOND QUEUE AREA (21 DECIMAL WORDS)

        .END    START
```


2.4.42 .RCTRLO

The .RCTRLO request ensures that the console terminal is able to print. Since CTRL/O struck while output is directed to the console terminal inhibits the output from printing until either another CTRL/O is struck or until the program resets the CTRL/O switch, a program that has a message that must appear at the console should reset the CTRL/O switch.

Macro Call: .RCTRLO

Errors:

None.

Example:

```
.TITLE  RCTRLO.MAC
;IN THIS EXAMPLE, THE USER PROGRAM FIRST CALLS THE CSI IN GENERAL MODE,
;THEN PROCESSES THE COMMAND. WHEN FINISHED, IT RETURNS TO THE CSI FOR
;ANOTHER COMMAND LINE. TO MAKE SURE THAT THE PROMPTING '*' TYPED BY
;THE CSI IS NOT INHIBITED BY A CTRL O IN EFFECT FROM THE LAST OPERATION,
;TERMINAL OUTPUT IS RE-ENABLED VIA A .RCTRLO COMMAND PRIOR TO THE
;CSI CALL.
        .MCALL  .RCTRLO,.CSIGEN,.EXIT

START:  .RCTRLO                 ;MAKE SURE TT OUTPUT IS
                                ;ENABLED
        .CSIGEN #DSPACE,#DEXT,#0 ;CALL CSI-IT WILL TYPE
                                ;"*"

                                ;PROCESS COMMAND

        JMP     START           ;GET NEXT COMMAND

DEXT:   0                       ;NO DEFAULT EXTENSIONS
        0
        0
        0
DSPACE: .=.+400         ;HANDLER SPACE

        .END    START
```

.RCVD/.RCVDC/.RCVDW

2.4.43 .RCVD/.RCVDC/.RCVDW (FB and XM Only)

There are three forms of the receive data request; these are used in conjunction with the .SDAT (Send Data) requests to allow a general data/message transfer system for communication between a foreground and a background job. .RCVD requests can be thought of as .READ requests where data transfer is not from a peripheral device but from the other job in the system. Additional queue elements should be allocated for buffered I/O operations in .RCVD and .RCVDC requests (see .QSET).

.RCVD

This request is used to receive data and continue execution. The request is posted and the issuing job continues execution. At some point when the job needs to have the transmitted message, an .MWAIT should be executed. This causes the job to be suspended until the message has been received.

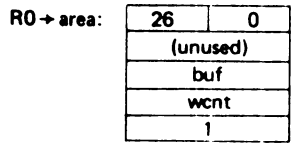
Macro Call: .RCVD area,buf,wcnt

where: area — is the address of a five-word EMT argument block.

buf — is the address of the buffer to which the message is to be sent.

wcnt — is the number of words to be transferred.

Request Format:



Word 0 (the first word) of the message buffer contains the number of words transmitted whenever the .RCVD is complete. Thus, the space allocated for the message should always be at least one word larger than the actual message size expected.

The word count is a variable number, and as such, the .SDAT/.RCVD combination can be used to transmit a few words or entire buffers. The .RCVD operation is only complete when a .SDAT is issued from the other job.

Programs using .RCVD/.SDAT must be carefully designed to either always transmit/receive data in a fixed format or to have the capability of handling variable formats. The messages are all processed in first in/first out order. Thus, the receiver must be certain it is receiving the message it actually wants.

Errors:

Code	Explanation
0	No other job exists in the system.

Example:

An example follows the .RCVDW section.

.RCVDC

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.

Macro Call: RCVDC area,buf,wcnt,crtn

where: area — is the address of a five-word EMT argument block.

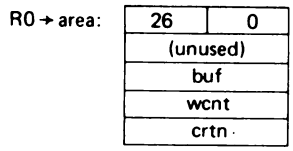
buf — is the address of the buffer to which the message is to be sent.

wcnt — is the number of words to be transmitted.

crtn — is the address of a completion routine to be entered.

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

Request Format:



Errors:

Code	Explanation
0	No other job exists in the system.

Example:

An example follows the .RCVDW section.

.RCVDW

.RCVDW is used to receive data and wait. A message request is posted and the job issuing the request is suspended until the other job sends a message to the issuing job. When the issuing job runs again, the message has been received, and word 0 of the buffer indicates the number of words transmitted.

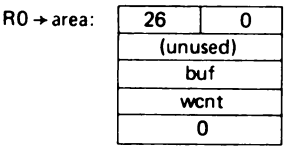
Macro Call: .RCVDW area,buf,wcnt

where: area — is the address of a five-word EMT argument block.

buf — is the address of the buffer to which the message is to be sent.

wcnt — is the number of words to be transmitted.

Request Format:



Errors:

Code	Explanation
0	No other job exists in the system.

Example:

```
.TITLE  RCVDMF.MAC
;THIS IS THE FOREGROUND JOB WHICH SENDS THE DATA TO THE RCVDM EXAMPLE.
        .MCALL  .SDATM,.PRINT,.EXIT

START:  MOV     #AREA,R5
        .SDATM  R5,#BUFR,#4     ;SEND 4 WORD FILE NAME
        BCC     1$
        .PRINT  #NJMSG
1$:     .EXIT
AREA:   .BLKW   5
BUFR:   .RAD50  /DK TEST  TMP/
NJMSG:  .ASCIZ  /NO A JOB/
        .EVEN
        .END    START

.TITLE  RCVD.MAC
;IN THIS EXAMPLE, THE RUNNING JOB RECEIVES A MESSAGE FROM THE
;SECOND JOB AND INTERPRETS IT AS THE DEVICE AND FILENAME OF A FILE
;TO BE OPENED AND USED. IN THIS CASE, THE MESSAGE WAS IN .RAD50 FORMAT,
;AND THE RECEIVING PROGRAM DID NOT USE THE TRANSMITTED LENGTH FOR ANY
;PURPOSE. THE ISSUING JOB IS SUSPENDED UNTIL THE INDICATED DATA
;IS TRANSMITTED. EITHER OF THE OTHER MODES COULD HAVE ALSO BEEN USED TO
;RECEIVE THE MESSAGE.
        .MCALL  .RCVDM,.PURGE,.LOOKUP,.EXIT,.PRINT

START:
        MOV     #AREA,R5        ;R5=EMT ARG. AREA
        .RCVDM  R5,#FILE,#4     ;REQUEST MESSAGE AND WAIT
        BCS     MERR            ;AN ERROR?
        .PURGE  #0              ;CLEAR CHANNEL 0
        .LOOKUP R5,#0,#FILE+2   ;LOOKUP INDICATED FILE
        BCS     LKERR           ;ERROR

        .EXIT

AREA:   .BLKW   10              ;LEAVE SPACE FOR SAFETY
FILE:   .BLKW   1               ;ACTUAL WORD COUNT IS HERE
        .BLKW   4               ;DEV:FILE.EXT ARE HERE

MERR:   .PRINT  #MMSG
        .EXIT
LKERR:  .PRINT  #LKMSG
        .EXIT
MMSG:   .ASCIZ  /MESSAGE ERROR/
LKMSG:  .ASCIZ  /LOOKUP ERROR/
        .EVEN
        .END    START
```


2.4.44 .READ/.READC/.READW

RT-11 provides three modes of I/O: .READ/.WRITE, .READC/.WRITC, and .READW/.WRITW.

In the case of .READ and .READC, additional queue elements should be allocated for buffered I/O operations (see .QSET).

NOTE

Upon return from any .READ, .READC or .READW programmed request, R0 contains the number of words requested if the read is from a sequential-access device (for example, paper tape). If the read is from a random-access device (disk or DECtape) R0 contains the actual number of words that will be read (.READ or .READC) or have been read (.READW). This number is less than the requested word count if an attempt is made to read past end-of-file, but a partial transfer is possible. In the case of a partial transfer, no error is indicated if a read request is shortened. Therefore, a program should always use the returned word count as the number of words available. For example, suppose a file is five blocks long (it has block numbers 0 to 4) and a request is issued to read 512 words, starting at block 4. Since 512 words is two blocks, and block 4 is the last block of the file, this is an attempt to read past end-of-file. The monitor detects this fact and shortens the request to 256 words. On return from the request, R0 contains 256 indicating that a partial transfer occurred. Also note that since the request is shortened to an exact number of blocks, a request for 256 words either succeeds or fails, but cannot be shortened.

An error is reported if a read is attempted with a block number that is beyond the end of file. The carry bit is set, and error code 0 appears in byte 52. No data is transferred, in this case. R0 contains a zero.

.READ

The .READ request transfers a specified number of words from the device associated with the specified channel to memory. The channel is associated with the device when a .LOOKUP or .ENTER request is executed. 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.

Macro Call: .READ area,chan,buf,wcnt,blk

where:

- area — is the address of a five-word EMT argument block.
- chan — is a channel number in the range 0-377 (octal).
- buf — is the address of the buffer to receive the data read.

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 blk=0, TT: issues an uparrow (^) prompt and LP: issues a form feed. (This is true for all .READ and .WRITE requests.)

The .READ and .READC requests perform the following operations:

1. Tell the monitor to do a read from the device and immediately return to the caller .

2. Execute as soon as all previous I/O requests to the device handlers have been completed. Note that a read from RK1: must wait for a previous read to RK0: to complete. This is a hardware restriction because the controller looks at all I/O operations sequentially.

3. Return read errors on the return from the .READ and .READC or the .WAIT request. Errors can occur on the read or on the wait, but only one error is returned. Therefore, the program must check for an error when the read is complete. The wait request returns an error, but it doesn't indicate which read caused the error.

4. During the .READ and .READC requests, the monitor keeps track of errors in the channel status word. If an error occurs before the monitor can return to the caller, the error is reported on the return from the read request with the carry bit set and the error values in R0. If the error occurs after return from the read request, the error is reported on return from the next .WAIT, or the next .READ/.READC. Some errors can be returned from .READ/.READC requests immediately, before any I/O operation takes place. One condition that causes an immediate error return is an attempt to read beyond end-of-file.

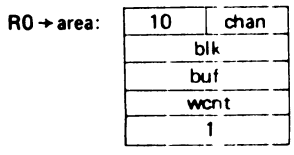
5. Errors reported on the return from the read request are:

   a. Non-existent device/unit

   b. Non-existent block

   c. In general, errors that don't require data tranfers but are controller errors or EOF errors

Request Format:



When the user program needs to access the data read on the specified channel, a .WAIT request should be issued. This ensures 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

Example:

Refer to the .WRITE/.WRITC/.WRITW examples.

.READC

The .READC request transfers a specified number of words from the indicated channel to memory. 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 an RTS PC is executed in the completion routine, control returns to the user program.

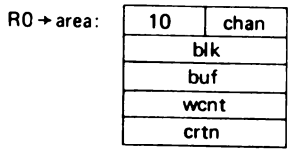
Macro Call: .READC area,chan,buf,wcnt,crtn,blk

where:

- area — is the address of a five-word EMT argument block.
- chan — is a channel number in the range 0-377 (octal).
- 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 (octal).
- 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.

When a completion routine is called, error or end-of-file information for a channel is not cleared. The next .WAIT or .READ/.READC on the channel (from either mainline code or a completion routine) produces an immediate return with the C bit set and the error code in byte 52.

Request Format:



When entering a .READC completion routine the following are true:

1. R0 contains the contents of the channel status word for the operation. If bit 0 of R0 is set, a hardware error occurred during the transfer; consequently, the data may not be reliable. The end-of-file bit may be set.

2. R1 contains the channel number of the operation. This is useful when the same completion function is to be used for several different transfers.

3. On a file-structured transfer, a shortened read is reported at the return from the .READC request, not when the completion routine is called.

4. Registers R0 and R1 can be used by the routine, but all other registers must be saved and restored. Data cannot be passed between the main program and completion routines in any register or on the stack.

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

Example:

Refer to the .WRITE/.WRITC/.WRITW examples.

.READW

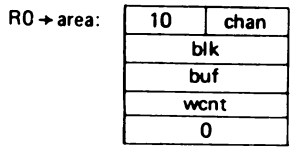
The .READW request transfers a specified number of words from the indicated channel to memory. Control returns to the user program when the .READW is complete and/or an error is detected.

Macro Call: .READW area,chan,buf,wcnt,blk

where:

- area — is the address of a five-word EMT argument block.
- chan — is a channel number in the range 0-377 (octal).
- 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.

Request Format:



On return from this call, a hardware error has occurred if the carry bit is set. If no error occurred, the data is in memory at the specified address. In an FB environment, the other job can be run while the issuing job is waiting for the I/O to complete.

Errors:

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

Example:

Refer to the .WRITE/.WRITC/.WRITW examples.

.RENAME

2.4.45 .RENAME

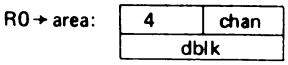
The .RENAME request causes an immediate change of name of the file specified and gives that file the current date in its directory entry. An error occurs if the channel specified is already open.

Macro Call: .RENAME area,chan,dblk

where:

- area — is the address of a two-word EMT argument block.
- chan — a channel number in the range 0-377 (octal)
- dblk — a block number specifying the relative file where an I/O transfer is to begin.

Request Format:



The dblk argument consists of two consecutive Radix-50 device and file specifications. For example:

```
            .RENAME   #AREA,#7,#DBLK      ;USE CHANNEL 7
            BCS       RNMERR              ;NOT FOUND
            .
            .
            .
DBLK:       .RAD50    /DT3/
            .RAD50    /OLDFIL/
            .RAD50    /MAC/
            .RAD50    /DT3/
            .RAD50    /NEWFIL/
            .RAD50    /MAC/
```

The first string represents the file to be renamed and the device where it is stored. The second represents the new file name. If a file with the same name as the new file name specified already exists on the indicated device, it is deleted. The second occurrence of the device name DT3 is necessary for proper operation, and should not be omitted. The specified channel is left inactive when the .RENAME is complete. .RENAME requires that the handler to be used be resident at the time the .RENAME request is made. If it is not, a monitor error occurs. Note that .RENAME is legal only on files on block-replaceable devices (disks and DECtape). In magtape operations, the handler returns an illegal operation code in byte 52 if a .RENAME request is attempted. (.RENAMEs to other devices are ignored.)

Errors:

Code	Explanation
0	Channel open
1	File not found
2	Illegal Operation

Example:

```
.TITLE  RENAME.MAC
;IN THIS EXAMPLE, THE FILE "DATA.TMP" ON DT0: IS RENAMED TO "DATA.001".
        .MCALL  .FETCH,.PRINT
        .MCALL  .EXIT,.RENAME

START:  .FETCH  #HSPACE,#NAMBLK ;GET HANDLER
        BCS     FERR            ;SOME ERROR
        .RENAME #AREA,#0,#NAMBLK ;DO THE RENAME
        BCS     RNMERR          ;ERROR
        .EXIT
FERR:   .PRINT  #FMSG
        .EXIT
RNMERR: .PRINT  #RNMSG
        .EXIT
AREA:   .BLKW   5               ;ROOM FOR ARGS.
NAMBLK: .RAD50  /DT0DATA  TMP/  ;OLD NAME
        .RAD50  /DT0DATA  001/  ;NEW NAME
FMSG:   .ASCIZ  /FETCH?/        ;ERROR MESSAGES
RNMSG:  .ASCIZ  /RENAME?/
        .EVEN
HSPACE:.

        .END    START
```

.REOPEN

2.4.46 .REOPEN

The .REOPEN request reassociates the specified channel with a file on which a .SAVESTATUS was performed. The .SAVESTATUS/.REOPEN combination is useful when a large number of files must be operated on at one time. As many files as are needed can be opened with .LOOKUP, and their status preserved with .SAVESTATUS. When data is required from a file, a .REOPEN enables the program to read from the file. The

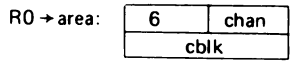
.REOPEN need not be done on the same channel as the original .LOOKUP and .SAVESTATUS.

Macro Call: .REOPEN area,chan,cblk

where:

area	is the address of a two-word EMT argument block.
chan	is a channel number in the range 0-377 (octal).
cblk	is the address of the five-word block where the channel status information was stored.

Request Format:



Errors:

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

Example:

Refer to the example following the description of .SAVESTATUS.

.SAVESTATUS

2.4.47 .SAVESTATUS

The .SAVESTATUS request stores five words of channel status information into a user-specified area of memory. These words contain all the information RT-11 requires to completely define a file. When a .SAVESTATUS is done, the data words are placed in memory, the specified channel is freed, and the file is closed. When the saved channel data is required, the .REOPEN request is used.

.SAVESTATUS can only be used if a file has been opened with .LOOKUP. If .ENTER was used, .SAVESTATUS is illegal and returns an error. Note that .SAVESTATUS is not legal only on magtape or cassette files.

The .SAVESTATUS/.REOPEN requests jointly are used to open many files on a limited number of channels or to allow all .LOOKUPs to be done at once to avoid USR swapping.

While the .SAVESTATUS/.REOPEN combination is very useful, care must be observed when using it. In particular, the following cases should be avoided:

1. If a .SAVESTATUS is performed and the same file is then deleted before it is reopened, it becomes available as an empty space that could be used by the .ENTER command. If this sequence occurs, the contents of the file supposedly saved changes.

2. Although the device handler for the required peripheral need not be in memory for execution of a .REOPEN, the handler must be in memory when a .READ or .WRITE is executed, or a fatal error is generated.

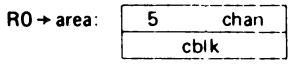
One of the more common uses of .SAVESTATUS and .REOPEN is to consolidate all directory access motion and code at one place in the program. All files necessary are opened and their status saved, then they are re-opened one at a time as needed. USR swapping can be minimized by .LOCKing in the USR, doing .LOOKUPs as needed, using .SAVESTATUS to save the file data, and then .UNLOCKing the USR. The user should be aware of the consequences of locking in the USR in a foreground/background environment. If the background job locks in the USR when the foreground job requires it, the foreground job is delayed until the background job unlocks the USR.

Macro Call: .SAVESTATUS area,chan,cblk

where:

area	is the address of a two-word EMT argument block.
chan	is a channel number in the range 0-377 (octal).
cblk	is the address of the five-word user memory block where the channel status information is to be stored.

Request Format:



Errors:

Code	Explanation
0	The channel specified is not currently associated with any files; that is, a previous .LOOKUP on the channel was never done.
1	The file was opened via .ENTER, or is a magtape or cassette file, and a .SAVESTATUS is illegal.

Example:

```
.TITLE  SAVEST,MAC
;ONE OF THE MORE COMMON USES OF .SAVESTATUS AND .REOPEN IS TO CONSOLIDATE
;ALL DIRECTORY ACCESS MOTION AND CODE AT ONE PLACE IN THE PROGRAM. ALL
;FILES NECESSARY ARE OPENED AND THEIR STATUS SAVED, THEN THEY ARE RE-OPENED
; ONE AT A TIME AS NEEDED. USR SWAPPING CAN BE MINIMIZED BY LOCKING IN THE
;USR, DOING .LOOKUP'S AS NEEDED, USING .SAVESTATUS TO SAVE THE FILE DATA,
;AND THEN UNLOCKING THE USR. IN THIS EXAMPLE THREE INPUT
;FILES ARE SPECIFIED IN THE COMMAND STRING;THESE ARE THEN PROCESSED ONE AT A TIME.
        .MCALL  .CSIGEN,.SAVESTATUS,.REOPEN
        .MCALL  .READ,.EXIT

START:  MOV     #AREA,R5
        .CSIGEN #DSPACE,#DEXT   ;GET INPUT FILES

        MOV     R0,BUFF         ;SAVE POINTER TO FREE CORE

        .SAVESTATUS R5,#3,#BLOCK1 ;SAVE FIRST INPUT FILE
        .SAVESTATUS R5,#4,#BLOCK2 ;SAVE SECOND FILE
        .SAVESTATUS R5,#5,#BLOCK3 ;SAVE THIRD FILE


        MOV     #BLOCK1,R4
PROCESS: .REOPEN R5,#0,R4       ;REOPEN FILE ON
                                ;CHANNEL 0

        .READ   R5,#0,BUFF,COUNT,BLOCK ;PROCESS FILE ON CHANNEL 0

DONE:   ADD     #12,R4          ;POINT TO NEXT SAVESTATUS BLOCK
        CMP     R4,#BLOCK3      ;LAST FILE PROCESSED?
        BLOS    PROCESS         ;NO - DO NEXT
        .EXIT

BLOCK1: .WORD   0,0,0,0,0       ;MEMORY BLOCKS FOR
BLOCK2: .WORD   0,0,0,0,0       ;SAVESTATUS INFORMATION
BLOCK3: .WORD   0,0,0,0,0
AREA:   .BLKW   10

BUFF:   .WORD   0
BLOCK:  .WORD   0
COUNT:  .WORD   256.

DEXT:   .WORD   0,0,0,0,0
DSPACE:.
        .END    START
```

.SCCA

2.4.48 .SCCA

The .SCCA request provides the following functions:

1. Inhibition of CTRL/C abort
2. Indication that a double CTRL/C was initiated at the keyboard.
3. Ability to distinguish between single and double CTRL/C commands.

This request intercepts and temporarily inhibits a console CTRL/C command, preventing the job from being aborted. CTRL/C characters are placed in the input ring buffer and are treated as normal control characters without specific system functions. The request requires a status word address that is used to report double CTRL/C input sequences. Bit 15 of the status word is set when consecutive CTRL/C characters are detected. The program must clear the bit.

There are two cautions to observe when using .SCCA. First, the request can cause CTRL/C to appear in the terminal input stream, and therefore the program must provide a way to handle it. Second, the request makes it impossible to terminate program loops from the console, and therefore it should be used only in thoroughly tested, reliable programs. When .SCCA causes an interminable program loop, the system must be halted and re-bootstrapped.

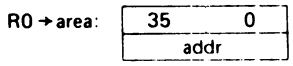
A .SCCA request with a status word address of zero disables the intercept and re-enables CTRL/C system action.

Macro Call: .SCCA area,addr

where: area — is the address of a two-word parameter block.

addr — is the address of a terminal status word (an address of 0 re-enables the CTRL/C command).

Request Format:



Errors:

None.

Example:

This example intercepts CTRL/C characters with the .SCCA request.

```
.TITLE  SCCA.MAC
        .MCALL  .SCCA,.PRINT,.TTYIN,.TTYOUT,    .EXIT

JSW     = 44                    ;JOB STATUS WORD
TTSPC$  = 10000                 ;SPECIAL MODE BIT
CTRL.C  = 3                     ;ASCII CTRL/C

START:: MOV     #SCCA,R1        ;R1 -> CTRL/C INDICATOR
        .SCCA   #AREA,R1        ;INTERCEPT CTRL/C
        BIS     #TTSPC$,@#JSW   ;SET SPECIAL MODE
        CLR     @R1             ;CLEAR INDICATOR
        .PRINT  #HELLO          ;TELL USER WE'RE RUNNING
10$:    .TTYIN                  ;READ A CHAR
        CMPB    R0,#CTRL.C      ;DID WE GET AN CTRL/C?
        BEQ     20$             ;BRANCH IF SO
        .TTYOUT                 ;WRITE A CHAR
        BR      10$             ;LOOP

20$:    .TTYOUT                 ;PRINT THE CTRL/C
        .PRINT  #CTRLC1         ;TELL USER ABOUT IT
30$:    TST     @R1             ;DOUBLE CTRL/C?
        BNE     40$             ;BRANCH IF SO
        .TTYIN                  ;READ A CHARACTER
        .TTYOUT                 ;WRITE A CHARACTER
        BR      30$             ;LOOP UNTIL DOUBLE CTRL/C STRUCK

40$:    .SCCA   #AREA,#0        ;LET MONITOR HANDLE CTRL/C'S
        .PRINT  #CTRLC2         ;INDICATE DOUBLE CTRL/C STRUCK
        .EXIT                   ; AND QUIT.

SCCA:   .BLKW   1               ;CTRL/C INTERRUPT WORD
AREA:   .BLKW   2               ;EMT AREA
HELLO:  .ASCII  \SCCA EXAMPLE PROGRAM\<15><12>
        .ASCIZ  \TYPE CTRL/C PLEASE\
CTRLC1: .ASCII  \THANK YOU, CTRL/C WAS DETECTED\<15><12>
        .ASCII  \TYPE <RET>, FOLLOWED BY DOUBLE \
        .ASCIZ  \CTRL/C TO ABORT PROGRAM\
        .EVEN
CTRLC2: .ASCIZ  \DOUBLE CTRL/C STRUCK\

        .END    START
```

.SDAT/.SDATC/.SDATW

2.4.49 .SDAT/.SDATC/.SDATW (FB and XM Only)

These requests are used in conjunction with the .RCVD/.RCVDW/.RCVDC calls to allow message transfers between a foreground job and a background job under the FB or XM monitors. .SDAT transfers can be considered similar to .WRITE requests in which data transfer is not to a peripheral, but from one job to another. Additional I/O queue elements should be allocated for buffered I/O operations in .SDAT and .SDATC requests (see .QSET).

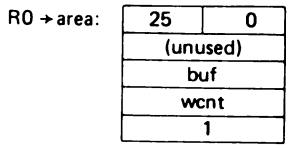
.SDAT

Macro Call: .SDAT area,buf,wcnt

where:

area	is the address of a five-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.

Request Format:



Errors:

Code	Explanation
0	No other job exists.

Example:

See the example following .SDATW.

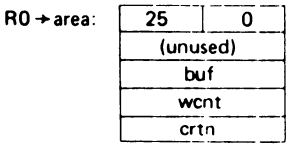
.SDATC

Macro Call: .SDATC area,buf,wcnt,crtn

where:

area	is the address of a five-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.

Request Format:



Errors:

Code	Explanation
0	No other job exists.

Example:

See the example following .SDATW.

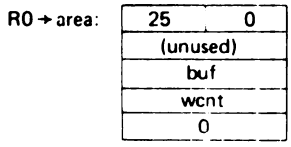
.SDATW

Macro Call: .SDATW area,buf,wcnt

where:

- area — is the address of a five-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.

Request Format:



Errors:

Code	Explanation
0	No other job exists.

Example:

```
        .TITLE  $DATWF.MAC
        ;THIS IS THE FOREGROUND HALF OF THE $DAT.MAC EXAMPLE.
        ;ONCE THE DATA IS RECEIVED, A MESSAGE INDICATING A POSITIVE ACKNOWLEDGEMENT
        ;IS SENT TO THE BACKGROUND.
                .MCALL  .$DATW,.RCVDW,.EXIT,.PRINT

        START:  MOV     #AREA,R5
                .RCVDW  R5,#BUFR1,#1,#176
                BCS     NJERR
                .$DATW  R5,#BUFR2,#2
                BCS     NJERR
                .EXIT
        AREA:   .BLKW   5
        NJERR:  .PRINT  #NJMSG
                .EXIT
        NJMSG:  .ASCIZ  /NO BACKGROUND JOB?/
                .EVEN
        BUFR1:  .BLKW   200
        BUFR2:  .ASCII  /YES./
                .EVEN
                .END    START

        .TITLE  $DAT.MAC
        ;IN THIS EXAMPLE, THE JOB FIRST SENDS A MESSAGE INTERROGATING THE OTHER
        ;JOB ABOUT THE STATUS OF AN OPERATION, AND THEN LOOKS FOR AN ACKNOWLEDGEMENT
        ;FROM THE JOB.
                .MCALL  .$DAT,.RCVD,.MWAIT,.PRINT,.EXIT

        START:
                MOV     #AREA,R5        ;SET UP EMT BLOCK
                .$DAT   R5,#$BUFF,#MLGTH ;ASK HIM A QUESTION
                BCS     NOJOB           ;NO OTHER JOB AROUND!

                                        ;MISCELLANEOUS PROCESSING

                .RCVD   R5,#BUFF2,#20.  ;RECEIVE 20 DECIMAL WORDS
                .MWAIT                  ;WAIT FOR ACKNOWLEDGE.
                MOV     #BUFF2+2,R1     ;POINT TO ACTUAL ANSWER.
                CMPB    (R1)+,#'Y       ;IS FIRST WORD Y FOR YES?
                BNE     PRNEG           ;NEGATIVE ACKNOWLEDGE
                .PRINT  #POSACK

                .EXIT

        PRNEG:  .PRINT  #NEGACK         ;NEGATIVE ON OUR INQUIRY
                .EXIT
        $BUFF:  .ASCII  /IS THE REQUIRED PROCESS GOING?/
        MLGTH=.-$BUFF
        BUFF2:  .WORD   0               ;ACTUAL LENGTH IS HERE
                .BLKW   20.             ;SPACE FOR 20. WORDS

        NOJOB:  .PRINT  #NJMSG
                .EXIT
        NEGACK: .ASCIZ  /NEGATIVE ACKNOWLEDGE/
        POSACK: .ASCIZ  /POSITIVE ACKNOWLEDGE/
        NJMSG:  .ASCIZ  /NO JOB/
                .EVEN
        AREA:   .BLKW   10.

                .END    START
```

.SETTOP

2.4.50 .SETTOP

The .SETTOP request allows the user program to request that a new address be specified as a program's upper limit. The monitor determines whether this address is legal and whether or not a memory swap is necessary when the USR is required. For instance, if the program specified an upper limit below the start address of USR (normally specified in offset 266), no swapping is necessary, as the USR is not overlaid. If .SETTOP from the background specifies a high limit greater than the address of the USR and a SET USR NOSWAP command has not been given, a memory swap is required. Section 2.2.5 gives details on determining where the USR is in memory and how to optimize the .SETTOP.

Careful .SETTOP usage provides a significant improvement in the performance of the user program. The following outline is a sample. Several of the system supplied programs use a similar approach.

A .SETTOP is done to the high limit of the code in a program before buffers or work areas are allocated. If the program is aborted, minimal writing of the user program occurs. However, the program is allowed to be restarted successfully.

A user command line is now read through .CSISPC or .GTLIN. An appropriate USR swap address is set in location 46. Successive .DSTATUS, .SETTOP and .FETCH requests are performed to load necessary device handlers. This attempts to keep the USR resident as long as possible during this procedure.

Buffers and work areas are allocated as needed, being sure to issue appropriate .SETTOP requests to account for their size. Frequently, a .SETTOP of #-2 is performed to request all available memory to be given to the program. This can be more useful than keeping the USR resident.

If the process has a well defined closing phase, another .SETTOP can be issued to cause the USR to become resident again to close files (the user should remember to set location 46 to zero if this is done, so that the USR again swaps in the normal area).

The program is now ready to cycle at the start of this procedure.

On return from .SETTOP, both R0 and the word at location 50 (octal) contain the highest memory address allocated for use. If the job requested an address higher than the highest address legal for the requesting job, the address returned is the highest legal address for the job rather than the requested address.

When doing a final exit from a program, the monitor writes the program to the file SWAP.SYS and then reads in the KMON. A .SETTOP #0 at exit time prevents the monitor from swapping out the program before reading in the KMON, thus saving time. This procedure is especially useful on a diskette system when indirect command files are used to run a sequence of programs.

Macro Call: .SETTOP addr

where: addr is the address of the highest word of the free area desired.

Notes:

1. A program should never do a .SETTOP and assume that its new upper limit is the address it requested. It must always examine the returned contents of R0 or location 50 to determine its actual high address.

2. It is imperative that the value returned in R0 or location 50 be used as the absolute upper limit. If this value is exceeded, vital parts of the monitor can be destroyed.

Errors:

None.

Example:

```
.TITLE  SETTOP.MAC
;THIS IS AN EXAMPLE IN TWO PARTS. THE FIRST INDICATES HOW A SMALL
;BACKGROUND JOB (I.E.,ONE WITH FREE SPACE BETWEEN ITSELF AND THE USR)
;CAN BE ASSURED OF RESERVING SPACE UP TO BUT NOT INCLUDING THE USR.
;THIS IN EFFECT GIVES THE JOB ALL THE SPACE IT CAN WITHOUT CAUSING
;THE USR TO BECOME NON-RESIDENT.
;THE SECOND PART INDICATES HOW TO ALWAYS RESERVE THE MAXIMUM AMOUNT
;OF SPACE BY MAKING THE USR NON-RESIDENT.
        .MCALL  .SETTOP,.EXIT,.GVAL

START:
;PART 1
RMON    = 54                    ;POINTER TO START OF RESIDENT
USRLOC  = 266                   ;OFFSET FROM RESIDENT TO POINTER
                                ;WHERE USR WILL START.

        .GVAL   #AREA,#USRLOC   ;R2 => USR
        TST     -(R0)           ;POINT TO HIGHEST WORD NOT IN USR
        .SETTOP                 ;AND ASK FOR IT
        MOV     R0,HICORE       ;R0 CONTAINS THE HIGH ADDRESS
                                ;THAT WAS RETURNED.

;PART 2
        .SETTOP #-2             ;IF WE ASK FOR A VALUE GREATER
                                ;THAN START OF RESIDENT, WE
                                ;WILL GET BACK THE ABSOLUTELY
                                ;HIGHEST USABLE ADDRESS.
        MOV     R0,HICORE       ;THAT IS OUR LIMIT NOW

        .EXIT
HICORE: .WORD   0
AREA:   .BLKW   2               ;EMT AREA BLOCK
        .END    START
```

If a SET USR NOSWAP command is executed, the USR cannot be made non-resident. In this case, in both 1 and 2 above, R0 would return a value just below the USR.

Caution should be used concerning technique 1, above. If the background program is so large that the USR is normally positioned over part of it, the high limit value returned by the .SETTOP can actually be lower than the program's original high limit determined at LINK time. The USR is then resident, with a portion of the user program destroyed. When the USR is resident, that portion of the user program is swapped out (no longer in memory) but it is reloaded automatically when the USR is no longer required.

.SFPA

2.4.51 .SFPA

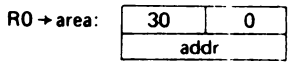
.SFPA allows users with floating point hardware to set trap addresses to be entered when a floating point exception occurs. If no user trap address is specified and a floating point (FP) exception occurs, a ?MON-F-FPU trap occurs, and the job is aborted.

Macro Call: .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.

Request Format:



Notes:

1. If the address argument is 0, user floating point routines are disabled and the fatal ?MON-F-FPU trap error is produced.

2. In the FB environment, an address value of 1 indicates that the FP registers should be switched when a context switch occurs, but no user traps are enabled. This allows both jobs to use the FP unit. An address of 1 to the SJ monitor is equivalent to an address of 0.

3. When the user routine is activated, it is necessary to re-execute an .SFPA request, as the monitor inhibit user traps when any one is serviced. It does this to prevent a possible infinite loop from being set up by repeated FP exceptions.

4. If the FP11 is being used, the instruction STST -(SP) is executed by the monitor before entering the user's trap routine. Thus, the trap routine must pop the two status words off the stack before doing an RTI. The program can tell if FP hardware is available by examining the configuration word in the monitor.

Errors:

None.

Example:

```
.TITLE  SFPA.MAC
;THIS EXAMPLE SETS UP A USER FP TRAP ADDRESS.
        .MCALL  .SFPA,.EXIT

START:

        .SFPA   #AREA,#FPTRAP

        .EXIT

FPTRAP:

        MOV     R0,-(SP)        ;R0 USED BY .SFPA
        .SFPA   #AREA,#FPTRAP
        MOV     (SP)+,R0        ;RESTORE R0
        RTT

AREA:   .BLKW   10
        .END    START
```

.SPFUN

2.4.52 .SPFUN

This request is used with cassette and magtape handlers to do device-dependent functions, such as rewind and backspace, on those devices. It can be used with diskettes and some disks to allow reading and writing of absolute sectors. This request can determine the size of a volume mounted in a particular device unit for RX02 diskettes, RK06/07 disks, and RL01 disks.

Macro Call: .SPFUN area,chan,func,buf,wcnt,blk[,crtn]

where:

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

chan — is a channel number in the range 0 to 377 (octal).

func — is the numerical code of the function to be performed.

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.

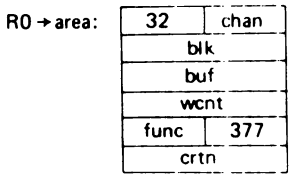
blk — is also defined in terms of the device handler associated with the specified channel and in terms of the specified special function code.

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
```

Request Format:



The chan, blk and wcnt arguments are the same as those defined for .READ/.WRITE requests. They are only required when doing a .WRITE with extended record gap to magnetic tape. If the crtn argument is left blank, the requested operation completes before control returns to the user program. CRTN=1 is equivalent to executing a .READ or .WRITE in that the function is initiated and returns immediately to the user program. A .WAIT on the channel ensures that the operation is completed. The crtn argument is a completion routine address to be entered when the operation is complete.

The available functions and their function codes are:

Function	MT	CT
Forward to last file		377
Forward to last block		376
Forward to next file		375
Forward to next block		374
Rewind to load point	373	373
Write file gap		372
Write EOF	377	
Forward 1 record	376	
Backspace 1 record	375	
Write	371	
Read	370	
Write with extended file gap	374	
Offline rewind	372	

Function	DX	DM	DY	DL
Read	377	377	377	377
Write	376	376	376	376
Write with deleted data mark	375		375	
Force a read by the handler of the bad block replacement table from block 1 of the disk.		374		374
Return device size		373	373	373

To use the .SPFUN request, the handler must be in memory and a channel associated with a file via a .LOOKUP request.

Other device specific .SPFUN requests are included in Chapter 1 with the appropriate device in the device handler section (section 1.4).

For the RK06/07 handler (DM), special function codes 377 and 376 require the buffer size to be one word larger than necessary for the data. The first word of the buffer contains the error information returned as a result of the .SPFUN request. The data transferred as a result of the read or write request is found in the second and following words of the buffer. The error codes and information are as follows:

Code	Meaning
100000	If the I/O operation is successful
10020	If a bad block is detected (BSE error)
100001	If an ECC error is corrected
100002	If an error recovered on retry
100004	If an error recovered through an offset retry
100010	If an error recovered after recalibration
1774xx	If an error did not recover

Errors:

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

Example:

```
.TITLE  SPFUN.MAC
;THE FOLLOWING EXAMPLE REWINDS A CASSETTE AND WRITES OUT A 256-WORD BUFFER
;AND THEN A FILE GAP.
        .MCALL  .FETCH,.LOOKUP,.SPFUN,.WRITW
        .MCALL  .EXIT,.PRINT,.WAIT,.CLOSE

START:
        .FETCH  #HSPC,#CT       ;GET A HANDLER
        BCS     FERR            ;FETCH ERROR
        .LOOKUP #AREA,#4,#CT    ;LOOK IT UP ON CHANNEL 4
        BCS     LKERR           ;LOOKUP ERROR
        .SPFUN  #AREA,#4,#373,#8 ;REWIND SYNCHRONOUSLY
        BCS     SPERR           ;AN ERROR OCCURRED.
        MOV     #3,R5           ;COUNT
                                ;BLOCK 0.
        .WRITW  #AREA,#4,#BUFF,#256.,BLK
        BCS     WTERR
        .SPFUN  #AREA,#4,#372,#0,,#1 ;ASYNCHRONOUS FILE GAP
        .PRINT  #DONE
        .WAIT   #4              ;WAIT FOR DONE
        .CLOSE  #4              ;CLOSE THE FILE
        .EXIT

AREA:   .BLKW   10
FERR:   .PRINT  #FMSG
        .EXIT
LKERR:  .PRINT  #LKMSG
        .EXIT
SPERR:  .PRINT  #SPMSG
        .EXIT
WTERR:  .PRINT  #WTMSG
        .EXIT
DONE:   .ASCIZ  /ALL DONE/
FMSG:   .ASCIZ  /FETCH?/
LKMSG:  .ASCIZ  /FILE?/
SPMSG:  .ASCIZ  /SPECIAL FUNCTION ERROR/
WTMSG:  .ASCIZ  /WRITE ERROR/
        .EVEN
CT:     .RAD50  /CT /
        .WORD   0,0,0
BUFF:   .BLKW   256.
BLK:    .WORD   0
HSPC=.
        .END    START
```

.SPND/.RSUM

2.4.53 .SPND/.RSUM (FB and XM Only)

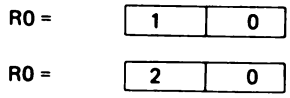
The .SPND/.RSUM requests allow a job to control execution of its mainstream code (that code which is not executing as a result of a completion routine). .SPND suspends the mainstream and allows only completion routines (for I/O and mark time requests) to run. .RSUM from one of the completion routines resumes the mainstream code. These functions enable a program to wait for a particular I/O or mark time request by suspending the mainstream and having the selected event's completion routine issue a .RSUM. This differs from the .WAIT request, which suspends the mainstream until all I/O operations on a specific channel have completed.

.SPND

Macro Calls: .SPND

.RSUM

Request Formats:



Notes:

1. The monitor maintains a suspension counter for each job. This counter is decremented by .SPND and incremented by .RSUM. A job is actually suspended only if this counter is negative. Thus, if a .RSUM is issued before a .SPND, the latter request returns immediately.

2. A program must issue an equal number of .SPNDs and .RSUMs.

3. A .RSUM request from the mainstream code increments the suspension counter.

4. A .SPND request from a completion routine decrements the suspension counter, but does not suspend the mainstream. If a completion routine does a .SPND, the mainstream continues until it also issues a .SPND, at which time it is suspended and requires two .RSUMs to proceed.

5. Since a .TWAIT is simulated in the monitor using suspend and resume, a .RSUM issued from a completion routine without a matching .SPND can cause the mainstream to continue past a timed wait before the entire time interval has elapsed.

6. A .SPND or .RSUM, like most other programmed requests, can be issued from within a user-written interrupt service routine if the .INTEN/.SYNCH sequence is followed. All notes referring to .SPND/.RSUM from a completion routine also apply to this case.

Errors:

None.

Example:

```
.TITLE  SPND.MAC
;IN THIS EXAMPLE, THE PROGRAM STARTS A NUMBER OF READ OPERATIONS
;AND SUSPENDS ITSELF UNTIL AT LEAST TWO OF THEM ARE COMPLETE.
        .MCALL  .SPND',.RSUM,.READC',.EXIT',.LOOKUP
        .MCALL  .PRINT,.WAIT

START:
        .LOOKUP #AREA,#2,#FILE2
        BCS     1$
        .LOOKUP #AREA,#3,#FILE3
        BCS     1$
        .LOOKUP #AREA,#4,#FILE4
        BCC     3$
1$:     .PRINT  #2$
        .EXIT
2$:     .ASCIZ  /LOOKUP ERROR/
        .EVEN
3$:


        MOV     #2,RSVCTR       ;WAIT FOR 2 COMPLETIONS
        MOV     #AREA,R5
        .READC  R5,#2,#BUF1,COUNT1,#CROUTN,BLOK1
        BCS     ERROR
        .READC  R5,#3,#BUF2,COUNT2,#CROUTN,BLOK2
        BCS     ERROR
        .READC  R5,#4,#BUF3,COUNT3,#CROUTN,BLOK3
        BCS     ERROR
        .SPND


        .WAIT   #2
        .WAIT   #3
        .WAIT   #4
        .EXIT

CROUTN: ASL     R1              ;DOUBLE CHANNEL # FOR INDEXING

        INC     DONEFL(R1)      ;R1=CHANNEL THAT IS DONE
                                ;SET A FLAG SAYING SO.
        ROR     R0              ;ANY ERRORS?
        ADC     ERRFLG(R1)      ;IF CARRY SET, SET ERROR FLAG FOR CHANNEL
        DEC     RSVCTR          ;ARE WE THE SECOND TO FINISH?
        BNE     1$              ;NO
        .RSUM                   ;YES, START UP
1$:     RTS     PC

ERROR:  .PRINT  #RDMSG
        .EXIT
RDMSG:  .ASCIZ  /READ ERROR/
        .EVEN
AREA:   .BLKW   10
RSVCTR: .WORD   0
COUNT1: .WORD   256.
COUNT2: .WORD   256.
COUNT3: .WORD   256.
BLOK1:  .WORD   0
BLOK2:  .WORD   0
BLOK3:  .WORD   0
FILE2:  .RAD50  /DK TEST2 TMP/
FILE3:  .RAD50  /DK TEST3 TMP/
FILE4:  .RAD50  /DK TEST4 TMP/
DONEFL: .WORD   0,0,0
ERRFLG: .WORD   0,0,0
BUF1:   .BLKW   256.
BUF2:   .BLKW   256.
BUF3:   .BLKW   256.

        .END    START
```

.SRESET

2.4.54 .SRESET

The .SRESET (software reset) request performs the following functions:

1. Dismisses any device handlers that were brought into memory via a .FETCH call. Handlers loaded via the keyboard monitor LOAD command remain resident, as does the system device handler.

2. Purges any currently open files. Files opened for output with .ENTER are never made permanent.

3. 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 (decimal) channels after a .SRESET is performed.

4. Resets the I/O queue to one element. A .QSET must be reissued to allocate extra queue elements.

5. Clears the completion queue of any completion routines.

Macro Call .SRESET

Errors:

None.

Example:

```
        .TITLE  SRESET.MAC
        ;IN THIS EXAMPLE, .SRESET IS USED PRIOR TO CALLING THE CSI TO ENSURE
        ;THAT ALL HANDLERS ARE REMOVED FROM MEMORY AND THE CSI IS STARTED WITH
        ;A FREE HANDLER AREA. IF THE .SRESET HAD NOT BEEN PERFORMED PRIOR TO THE
        ;SECOND CALL OF CSIGEN, IT IS POSSIBLE THAT THE SECOND COMMAND STRING
        ;WOULD LOAD A HANDLER OVER ONE THAT THE MONITOR THOUGHT WAS RESIDENT FROM
        ;THE FIRST COMMAND LINE.
                .MCALL  .CSIGEN,.SRESET,.EXIT

        START:  .CSIGEN #DSPACE,#DEXT,#0 ;GET COMMAND STRING
                MOV     R0,BUFFER       ;R0 POINTS TO FREE MEMORY


        DONE:   .SRESET                 ;RELEASE HANDLERS, DELETE
                                        ;TENTATIVE FILES
                BR      START           ;AND REPEAT PROGRAM.

        DEXT:   .WORD   0,0,0,0         ;NO DEFAULT EXTENSIONS
        BUFFER: 0
        DSPACE=.                        ;START OF HANDLER AREA.


                .END    START
```

.SYNCH

2.4.55 .SYNCH

This macro call enables the user program to perform monitor programmed requests from within an interrupt service routine. Code following the .SYNCH call executes at processor priority 0 and in the issuing job's context. Unless a .SYNCH is used, issuing programmed requests from interrupt routines is not supported by the system and should not be performed. .SYNCH, like .INTEN, is not an EMT monitor request, but rather a subroutine call to the monitor.

Macro Call: .SYNCH area[,pic]

where: area is the address of a seven-word block that the user must set aside for use by .SYNCH. Note that the argument, area, represents a special seven-word block used by .SYNCH. 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 1	RT-11 maintains this word; its contents should not be altered by the user.
Word 2	The current job's number. This can be obtained by a .GTJB call.
Word 3	Unused.
Word 4	Unused.
Word 5	R0 argument. When a successful return is made from .SYNCH, R0 contains this argument.
Word 6	Must be -1.
Word 7	Must be 0.

pic is an optional argument that enables the .SYNCH macro to produce position independent code for use by device drivers.

Note:

.SYNCH assumes that the user has not pushed anything on the stack between the .INTEN and .SYNCH calls. This rule must be observed for proper operation.

Errors:

The monitor returns to the location immediately following the .SYNCH if the .SYNCH was rejected. The routine is still unable to issue programmed requests, and R4 and R5 are available for use. Errors returned are due to one of the following causes:

1. Another .SYNCH that specified the same seven-word block is still pending.

2. An illegal job number was specified in the second word of the block. The only currently legal job numbers are 0 and 2.

3. If the job has been aborted or for some reason is no longer running, the .SYNCH fails.

Normal return is to the word after the error return with the routine in user state and thus allowed to issue programmed requests. R0 contains the argument that was in word 5 of the block. R0 and R1 are free to be used without having to be saved. R4 and R5 are not free, and do not contain the same information they contained before the .SYNCH request. A long time can elapse before the program returns from a .SYNCH request since all interrupts must be serviced before the main program can continue. Exit from the routine should be done via an RTS PC.

Example:

```
.TITLE  SYNCH.MAC
        .MCALL  .GTJB,.INTEN,.WRITC,.SYNCH,.EXIT,.PRINT
START:  MOV     #JOB,R5             ;OUTPUT OF .GTJB GOES HERE
        .GTJB   #AREA,R5            ;GET JOB NUMBER
        MOV     (R5),SYNBLK+2       ;STORE THE JOB NUMBER INTO SYNCH BLOCK
                                    ;IN HERE WE SET UP INTERRUPT
                                    ;PROCESSING, AND START UP THE
                                    ;INTERRUPTING DEVICE.

INTRPT: .INTEN  5                   ;GO INTO SYSTEM STATE
                                    ;RUN AT LEVEL FIVE
                                    ;INTERRUPT PROCESSING --
                                    ;NOTHING CAN GO ON STACK
        .SYNCH  #SYNBLK             ;TIME TO WRITE A BUFFER
        BR      SYNFAIL             ;SYNCH BLOCK IN USE

;RETURN HERE AT PRIORITY 0.  NOTE: .SYNCH DOES RTI

        .WRITC  #AREA,CHAN,BUFF,WCNT,#CRTN1,BLK
                                    ;WRITE A BUFFER
        BCS     WTFAIL              ;FAILED SOMEHOW

                                    ;RE-INITIALIZE FOR MORE
        RTS     PC                  ;INTERRUPTS AND EXIT
SYNBLK: .WORD   0
        .WORD   0                   ;JOB NUMBER
        .WORD   0
        .WORD   0
        .WORD   5                   ;R0 CONTAINS 5 ON SCCESSFUL
                                    ;SYNCH
        .WORD   -1,0                ;SET UP FOR MONITOR
SYNFAIL:


WTFAIL: MOV     #MSG2,R0
TELLEM: .PRINT
        .EXIT
MSG2:   .ASCIZ  /WRITE FAIL/
        .EVEN
AREA:   .BLKW   5
JOB:    .BLKW   5                   ;BUFFER FOR .GTJB
CRTN1:

        RTS     PC
CHAN:   .WORD   0                   ;CHANNEL #
BUFF:   .BLKW   256.
WCNT:   .WORD   0                   ;WORD COUNT
BLK:    .WORD   0                   ;BLOCK #

        .END    START
```

.TLOCK

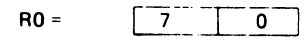
2.4.56 .TLOCK (FB and XM Only)

.TLOCK is used in an FB 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 SJ monitor). However, if a job attempts to .LOCK the USR while the other 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.

The .TLOCK request allows the job to continue running, with only one sub-job affected. With a .LOCK request, all sub-jobs are automatically suspended, and the other job in the system runs.

Macro Call: .TLOCK

Request Format:



Errors:

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

Example:

```
.TITLE  TLOCK.MAC
;IN THIS EXAMPLE, THE USER PROGRAM NEEDS THE USR FOR A SUB-JOB IT IS
;EXECUTING. IF IT FAILS TO GET THE USR IT SUSPENDS THAT SUB-JOB AND
;RUNS ANOTHER SUB-JOB. THIS TYPE OF PROCEDURE IS USEFUL TO SCHEDULE SEVERAL
;SUB-JOBS WITHIN A BACKGROUND OR FOREGROUND PROGRAM.
        .MCALL  .TLOCK,.LOOKUP,.UNLOCK,.EXIT,.PRINT

START:


        .TLOCK                      ;GET THE USR
        BCS     SUSPND              ;FAILED, SUSPEND SUB-JOB
        .LOOKUP #AREA,#4,#J1NAM ;LOOKUP A FILE
        BCS     LKERR
        .UNLOCK                     ;RELEASE USR


        .EXIT

SUSPND: JSR     PC,SPSJOB           ;SUSPEND SUB-JOB
        JSR     PC,SCHED            ;AND SCHEDULE NEXT USER
        BR      START               ;LOOP

AREA:   .BLKW   10
J1NAM:  .RAD50  /DK TEST1 TMP/
LKERR:  .PRINT  #LKMSG
        .EXIT
LKMSG:  .ASCIZ  /LOOKUP ERROR/
        .EVEN
SPSJOB: RTS     PC
SCHED:  RTS     PC

        .END    START
```

.TRPSET

2.4.57 .TRPSET

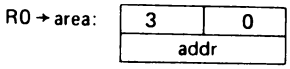
.TRPSET allows the user job to intercept traps to 4 and 10 instead of having the job aborted with a ?MON-F-Trap to 4 or ?MON-F-Trap to 10 message. If .TRPSET is in effect when an error trap occurs, the user-specified routine is entered. The sense of the carry bit on entry to the routine determines which 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 to 4 can also be caused by user stack overflow on some processors. These traps are not intercepted by the .TRPSET request but they do cause job abort and a printout of the message ?MON-F-Stack overflow (in the SJ monitor) or ?MON-F-Trap to 4 (in the FB and XM monitors).

Macro Call: .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, the user's trap interception is disabled.

Request Format:



Notes:

It is necessary to reissue a .TRPSET request whenever an error trap occurs and the user routine is entered. The monitor inhibits servicing user traps prior to entering the first user trap routine. Thus, if a trap should occur from within the user's trap routine, an error message is generated and the job is aborted. The last operation the user routine should perform before an RTI is to reissue the .TRPSET request.

In the XM monitor, traps dispatched to a user program by .TRPSET execute in user mode. They appear as interrupts of the user program by a synchronous trap operation. Programs that intercept errors traps by trying to steal the trap vectors must be carefully designed to handle two cases accurately: programs that are virtual jobs and programs that are privileged jobs.

If the program is a virtual job, the stolen vector is in user virtual space that is not mapped to kernel vector space. The proper method is to use .TRPSET; otherwise interception attempts fail and the monitor continues to handle traps to 4 and 10.

If the program is a privileged job, it is mapped to the kernel vector page. The user can steal the error trap vectors from the monitor, but the benefits of doing so must be carefully evaluated in each case. Trap routines run in the mapping mode specified by bits 14 and 15 of the trap vector PS word. With both bits set to 0, kernel mode is set. However, kernel mapping is not always equivalent to user mapping, particularly when extended memory is being used. With both PS word bits set to 1, user mode is set, and the trap routine executes in user mapping.

Errors:

None.

Example:

```
.TITLE  TRPSET,MAC
;IN THIS EXAMPLE, A USER TRAP ROUTINE IS SET AND, WHEN THE TRAP OCCURS,
;THE USER TRAP ROUTINE PRINTS AN APPROPRIATE ERROR MESSAGE.
        .MCALL  .TRPSET,.EXIT,.PRINT

START:

        .TRPSET #AREA,#TRPLOC
        MOV     #101,R0         ;SET TO PRODUCE A TRAP
        TST     (R0)+           ;THIS WILL TRAP TO 4.
        .WORD   67              ;THIS WILL TRAP TO 10.
        .EXIT


TRPLOC: MOV     R0,-(SP)        ;R0 USED BY EMTS
        BCS     1$              ;C SET = TRAP TO 10.
        .PRINT  #TRP4           ;TRAP TO 4
        BR      2$
1$:     .PRINT  #TRP10          ;TRAP TO 10
2$:     .TRPSET #AREA,#TRPLOC   ;RESET TRAP ADDRESS
        MOV     (SP)+,R0        ;RESTORE R0
        RTI


AREA:   .BLKW   10
TRP4:   .ASCIZ  /TRAP TO 4/
TRP10:  .ASCIZ  /TRAP TO 10/
        .EVEN

        .END    START
```


2.4.58 .TTYIN/TTINR

These requests are used to transfer characters 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, or R0 and other locations.

The expansion of .TTYIN is:

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

The expansion of .TTINR is:

```
EMT 340
```

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.

If the carry bit is set when execution of the .TTINR request is completed, it indicates that no character was available; the user has not yet typed a valid line. Under the FB or XM monitor, .TTINR does not return the carry bit set unless bit 6 of the job status word (JSW) was on when the request was issued (see below).

There are two modes of doing console terminal input. This is governed by bit 12 of the job status word. If bit 12 = 0, normal I/O is performed. In this mode, the following conditions apply:

1. The monitor echoes all characters typed.

2. CTRL/U and the DELETE key perform line deletion and character deletion, respectively.

3. A carriage return, line feed, CTRL/Z, or CTRL/C must be struck before characters on the current line are available to the program. When carriage return is typed, characters on the line typed are passed one-by-one to the user program; both carriage return and line feed are passed to the program.

If bit 12 = 1, the console is in special mode. The effects are:

1. The monitor does not echo characters typed except for CTRL/C and CTRL/O.

2. CTRL/U and the DELETE key do not perform special functions.

3. Characters are immediately available to the program.

4. No ALTMODE conversion is done.

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. Bit 12 in the JSW must be set by the user program. This bit is cleared when the program terminates.

CTRL/F and CTRL/B are not affected by the setting of bit 12. The monitor always acts on these characters (unless the SET TT NOFB command is issued).

CTRL/S and CTRL/Q are intercepted by the monitor (unless, under the FB or XM monitor, the SET TT NOPAGE command is issued).

Under the FB or XM monitor, if a terminal input request is made and no character is available, job execution is blocked until a character is ready. This is true for both .TTYIN and .TTINR, and for both normal and special modes. If a program requires execution to continue and the carry bit to be returned, it must set bit 6 of the JSW (location 44) before the .TTINR request. Bit 6 is cleared when a program terminates.

NOTE

The .TTYIN request does not support (track) indirect files. If this function is desired, the .GTLIN request must be used. .TTYIN cannot get a character from an indirect command file.

Macro Calls: .TTYIN char

.TTINR

where: char is a pointer to the location where the character in R0 is to be stored. If the character is specified, the character is in R0 and the address is pointed to by the character. If the character is not specified, the character is in R0.

Errors:

Code	Explanation
0	No characters available in ring buffer.

Example:

Refer to the example following the description of .TTYOUT/.TTOUTR.

2.4.59 .TTYOUT/.TTOUTR

These requests cause a character to be transmitted from R0 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 in R0 is not output.

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. Under the FB or XM monitor, .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 bit 6 of the job status word (location 44) before issuing the request.

The .TTINR and .TTOUTR requests have been supplied as a help to those users who do not need to 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.

NOTE

If a foreground job leaves bit 6 set in the JSW, any further foreground .TTYIN or .TTYOUT requests cause the system to lock out the background. Note also that each job in the foreground/background environment has its own JSW, and therefore can be in different terminal modes independently of the other job.

Macro Calls: .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.

Errors:

Code	Explanation
0	Output ring buffer full.

Example:

```
.TITLE  TTINR.MAC
;SIMILAR TO TTYIN.MAC, BUT RATHER THAN WAITING FOR THE USER TO TYPE
;SOMETHING AT INLOOP OR WAIT FOR THE OUTPUT BUFFER TO HAVE AVAILABLE
;SPACE AT OUTLOOP, THE ROUTINE HAS BEEN RECODED USING .TTINR AND .TTOUTR.
        .MCALL  .TTYIN,.TTYOUT
        .MCALL  .TTINR,.TTOUTR,.EXIT
JSW     = 44
START:  MOV     #BUFFER,R1      ;POINT R1 TO BUFFER
        CLR     R2              ;CLEAR CHARACTER COUNT
        BIS     #100,@#JSW      ;WE REALLY WANT CARRY SET
INLOOP: .TTINR                  ;GET CHAR FROM TERMINAL
        BCS     NOCHAR          ;NONE AVAILABLE
CHKIN:  MOVB    R0,(R1)+        ;PUT CHAR IN BUFFER
        INC     R2              ;INCREASE COUNT
        CMPB    R0,#12          ;WAS LAST CHAR = LF?
        BNE     INLOOP          ;NO-GET NEXT
        MOV     #BUFFER,R1      ;YES-POINT R1 TO BUFFER
OUTLOOP: MOVB   (R1),R0         ;PUT CHAR IN R0
        .TTOUTR                 ;TYPE IT
        BCS     NOROOM          ;NO ROOM IN OUTPUT BUFFER
CHROUT: DEC     R2              ;DECREASE COUNT
        BEQ     START           ;DONE IF COUNT=0
        INC     R1              ;BUMP BUFFER POINTER
        BR      OUTLOOP         ;AND TYPE NEXT
NOCHAR:



        .TTINR                  ;PERIODIC CHECK FOR
                                ;CHARACTER AVAILABILITY
        BCC     CHKIN           ;GOT ONE

                .
                .
                .

        BR      NOCHAR

NOROOM:

        MOVB    (R1),R0         ;PERIODIC ATTEMPT TO TYPE
                                ;CHARACTER
        .TTOUTR
        BCC     CHROUT          ;SUCCESSFUL
                                ;CODE TO BE EXECUTED WHILE WAITING


        .REPT   10
        NOP
        .ENDR

        BIC     #100,@#JSW      ;MUST CLEAR THIS BIT
                                ;SO HANG WHILE
                                ;WAITING FOR ROOM.
        .TTYOUT (R1)            ;PUT CHAR
        BIS     #100,@#JSW      ;RESTORE NO-WAIT
        BR      CHROUT



BUFFER: .BLKB   100.*
        .END    START
```

.TWAIT

2.4.60 .TWAIT (FB and XM Only)

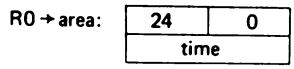
The .TWAIT request suspends the user's job for an indicated length of time. .TWAIT requires a queue element, and thus, should be considered when the .QSET request is executed.

Macro call: .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.

Request Format:



Notes:

Since a .TWAIT is simulated in the monitor using suspend and resume, a .RSUM issued from a completion routine without a matching .SPND can cause the mainstream to continue past a timed wait before the entire time interval has elapsed. In addition, a .TWAIT issued within a completion routine is ignored by the monitor, since it would block the job from ever running again.

The unit of time for this request is clock ticks, which can be 50 Hz or 60 Hz, depending on the local power supply. This must be kept in mind when the time interval is specified.

Errors:

Code	Explanation
0	No queue element was available.