

Table of contents

2-	1	RTEMT	-- Real-time emt entry point
3-	1	RTCVAD	-- Convert virtual address to physical address
4-	1	RTVMAP	-- Map virtual address to physical region
5-	1	RTFREZ	-- Lock job in memory without repositioning
5-	11	RTUNLK	-- Unlock job from memory
6-	1	RTPEEK	-- Peek at I/O page
6-	15	RTPOKE	-- Poke into I/O page
6-	27	RTBIS	-- Bit set into I/O page
6-	39	RTBIC	-- Bit clear into I/O page
7-	1	RTIOMP	-- Map par7 to I/O page
7-	11	RTRMMP	-- Map par7 to simulated Rmon
8-	1	.SPND	-- Suspend job's execution
8-	27	.RSUM	-- Resume job's execution
9-	1	RTCMP	-- Schedule a completion routine
10-	1	RTEXJB	-- Gain exclusive use of system
10-	17	RTALJB	-- Release exclusive access claim to system
11-	1	RTSPL	-- Set processor priority level
12-	1	RTCVEC	-- Connect completion routine to interrupt vector
13-	1	RTRVEC	-- Release interrupt vector
14-	1	RTDEV	-- .DEVICE Emt
15-	1	RTSTOP	-- Real-time cleanup at end of job execution
17-	1	** Subroutines **	
17-	2	PRVRT	-- Determine if job has real-time privilege
17-	13	PRVMEM	-- Check for access authorization to phys memory
18-	1	SRCVEC	-- Search for vector control block

```

1          .TITLE  TSRTX -- Non-resident Real-Time Support Module
2          .ENABL  LC
3          .DSABL  CBL
4          ;-----
5          ; TSRTX is the non-resident portion of the TSX-Plus real-time support
6          ; code.
7          ;
8          ; Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985.
9          ; S&H Computer Systems, Inc.  Nashville, Tennessee
10         ; All rights reserved.
11         ;
12 000000          .CSECT  TSRTX
13 000000  071670  TSRTX:  .RAD50  /RTX/          ;Overlay id
14         ;
15         ; Macro definitions
16         ;
17         ; Macro to call a routine in another system overlay region.
18         ;
19         .MACRO  OCALL  ENTADD
20         .IF     B,ENTADD
21         .ERROR  ;OCALL without entry address
22         .ENDC
23         CALL   OVRHC
24         .WORD  ENTADD)
25         .ENDM  OCALL
26         ;
27         ; Global definitions
28         ;
29         .GLOBL RTSPND, RTRSUM
30         .GLOBL RTSTOP, RTDEV, RTEMT
31         ;
32         ; Global references
33         ;
34         .GLOBL LSW2, SETERR, EMTBLK, BADEMT, LPARBS, EMTXIT
35         .GLOBL $IOMAP, LSW6, SETMAP, VC#JOB, VC#VEC, VC#RTN
36         .GLOBL UEXINT, CQ#JOB, CQ#RO, CQ#RTN, QCOMPL
37         .GLOBL URO, VC#PRI, ENTPS, INTPRI, RTLOCK
38         .GLOBL S$TWFN, LBSPRI, $MLOCK, PO$LOK
39         .GLOBL OVRHC, UEXRTN, VUXIFL
40         .GLOBL VCBBAS, VCBEND, VC##SZ, PO$MEM, PO$RT, PRIVCO
41         .GLOBL LSPND, CURCP, S$SPND, CHKABT
42         .GLOBL QHDSPN, DEVL, DEVLS, CQ$PRI, CQ$PA5, CQ$RNS
43         .GLOBL DOSCHD, EXCJOB, CINFLG
44         .GLOBL RPAR, RPDR, VF$DIR, VC$FLG, VF$DET
45         .GLOBL UPARO, UPDRO, CUPARO, CUPDRO
46         .GLOBL UPMODE, UPMODE, KPAR5, CQ$CP, CP$STD, CP$RT
47         .GLOBL CUPARO, CUPDRO, CQ$R1
48         .GLOBL GETQ, CORUSR, MAXPRI, S$RT, VPRIHI
49         ;
50         ;-----
51         ; MACROS TO ENABLE AND DISABLE INTERRUPTS.
52         ;
53         PS      =      177776          ;PROCESSOR STATUS WORD
54         .MACRO  DISABL          ;DISABLE INTERRUPTS
55         BIS    #340, @#PS
56         .ENDM  DISABL
57         ;

```

```
58          .MACRO ENABL          ;ENABLE INTERRUPTS
59          BIC      INTPR1,@#P5
60          .ENDM ENABL
```

RTEMT -- Real-time emt entry point

```

1          .SBTTL RTEMT -- Real-time emt entry point
2          ;-----
3          ; RTEMT is jumped to from ISEMT whenever one of the real-time emt's
4          ; is executed.
5          ; The general form of a real-time emt is
6          ;
7          ;     .BYTE   sub-function,140
8          ;     .WORD   Arg1    ;(optional)
9          ;     .WORD   Arg2    ;(optional)
10         ;     .WORD   Arg3    ;(optional)
11         ;
12         ; Where sub-function is a code that indicates which real-time
13         ; function is to be performed. (See vector at end of this routine)
14         ;
15         ; See if user is authorized to use real-time emt's.
16         ;
17 000002 RTEMT:
18         ;
19         ; Get sub-function code from arg block and jump to processing routine
20         ;
21 000002 116705 0000000 1$:   MOVB   EMTBLK,R5      ;GET SUB-FUNCTION CODE
22 000006 006305          ASL    R5              ;CONVERT TO WORD TABLE INDEX
23 000010 020527 000042   CMP    R5,#MXRTFN      ;IS IT VALID SUB-FUNCTION CODE?
24 000014 101402          BLOS   2$              ;BR IF OK
25 000016 000167 0000000   JMP    BADEMT         ;INVALID SUB-FUNCTION VALUE
26 000022 000175 000026' 2$:   JMP    @RTEMTV(R5)   ;JUMP TO PROCESSING ROUTINE FOR EMT
27         ;
28         ;-----
29         ; Define sub-function jump vector.
30         ;
31 000026 000072' RTEMTV: .WORD   RTCVAD      ;00 - Convert virtual address to physical
32 000030 000456'          .WORD   RTPEEK      ;01 - Peek at I/O page
33 000032 000472'          .WORD   RTPOKE      ;02 - Poke into I/O page
34 000034 000506'          .WORD   RTBIS       ;03 - Bit set into I/O page
35 000036 000522'          .WORD   RTBIC       ;04 - Bit clear into I/O page
36 000040 000540'          .WORD   RTIOMP      ;05 - Map virtual PAR6 to I/O page
37 000042 000560'          .WORD   RTRMMP      ;06 - Map virtual PAR6 to simulated RMON
38 000044 0000000          .WORD   RTLOCK      ;07 - Lock job in low memory
39 000046 000444'          .WORD   RTUNLK      ;10 - Unlock job from memory
40 000050 001144'          .WORD   RTCVEC      ;11 - Connect completion routine to interrupt
41 000052 001274'          .WORD   RTRVEC      ;12 - Disconnect completion rtn from interrupt
42 000054 000420'          .WORD   RTFREZ      ;13 - Lock job in memory without repositioning
43 000056 001024'          .WORD   RTEXJB      ;14 - Gain exclusive use of system
44 000060 001040'          .WORD   RTALJB      ;15 - Release exclusive use of system
45 000062 001054'          .WORD   RTSPL       ;16 - Set processor priority level level
46 000064 000172'          .WORD   RTVMAP      ;17 - Map virtual region to physical region
47 000066 001112'          .WORD   RTDVEC      ;20 - Connect direct interrupt service routine
48 000070 000660'          .WORD   RTCMPL      ;21 - Schedule completion routine
49         ;
50         000042 MXRTFN = .-RTEMTV-2 ;2 * Highest legal subfunction number

```

RTCVAD -- Convert virtual address to physical address

```

1          .SBTTL  RTCVAD -- Convert virtual address to physical address
2          ;-----
3          ; The RTCVAD emt is used to convert a 16-bit virtual address to a
4          ; 22-bit physical address.
5          ;
6          ; Inputs:
7          ;   Arg1 = Virtual address
8          ;   Arg2 = Address of 2-word block to receive 22-bit address result
9          ;
10         ; Outputs:
11         ;   Result buffer in user's area receives 22-bit physical address
12         ;   low-order 16-bits stored in 1st word and high-order 2-bits positioned
13         ;   in bit positions 4-5 are stored in 2nd word.
14         ;
15 000072 016700 0000020 RTCVAD: MOV     EMTBLK+2,R0    ;GET VIRTUAL ADDRESS
16 000076 010003          MOV     R0,R3
17 000100 005002          CLR     R2                ;SET UP FOR SHIFT
18 000102 073227 000003  ASHC   #3,R2        ;GET 3 HIGH-ORDER BITS OF VIRTUAL ADDRESS
19 000106 006302          ASL   R2                ;*2 TO CONVERT TO WORD TABLE INDEX
20 000110 005762 0000000  TST   RPDR(R2)        ;IS THIS REGION SPECIALLY MAPPED?
21 000114 001405          BEQ   1$                ;BR IF NOT
22 000116 016205 0000000  MOV   RPAR(R2),R5     ;GET PHYSICAL ADDRESS FOR THIS PAR
23 000122 042700 160000  BIC   #160000,R0      ;REMOVE PAR # FROM VIRTUAL ADDRESS
24 000126 000402          BR    2$
25 000130 016105 0000000  1$:  MOV   LPARBS(R1),R5 ;GET BASE 64-BYTE BLOCK NUMBER OF JOB REGION
26 000134 005004          2$:  CLR   R4                ;CLEAR HIGH-ORDER WORD
27 000136 073427 000006  ASHC   #6,R4        ;CONVERT TO 22-BIT PHYSICAL ADDR OF JOB BASE
28 000142 060005          ADD   R0,R5        ;ADD VIRTUAL ADDRESS
29 000144 005504          ADC   R4                ;PROPOGATE CARRY
30 000146 072427 000004  ASH   #4,R4        ;PUT 2 HIGH-ORDER BITS IN BIT POSITIONS 4-5
31 000152 016703 0000040  MOV   EMTBLK+4,R3    ;GET ADDRESS OF BUFFER WHERE RESULT GOES
32 000156 010546          MOV   R5,-(SP)      ;STACK LOW-ORDER PART OF RESULT
33 000160 106623          MTPD  (R3)+        ;STORE INTO USER'S BUFFER
34 000162 010446          MOV   R4,-(SP)      ;STACK HIGH-ORDER PART OF RESULT
35 000164 106613          MTPD  (R3)         ;STORE INTO USER'S BUFFER
36 000166 000167 0000000  JMP   EMTXIT        ;FINISHED

```

RTVMAP -- Map virtual address to physical region

```

1          .SBTTL  RTVMAP -- Map virtual address to physical region
2          ;-----
3          ; This EMT is used to map a virtual region of memory to a specified
4          ; physical region.
5          ;
6          ; .BYTE 17,140
7          ; .WORD Base PAR # (0-7)
8          ; .WORD Physical base (64-byte block #)
9          ; .WORD Region length (# 64-byte blocks)
10         ; .BYTE Access control (0==>read only, 1==>read/write)
11         ; .BYTE Cache control (0==>enable caching, 1==>bypass cache)
12         ;
13 000172 004767 001534 RTVMAP: CALL  PRVMEM          ; Can we access physical memory?
14 000176 016705 0000060      MOV    EMTBLK+6,R5      ; GET SIZE OF REGION
15 000202 001016              BNE    10$              ; BR IF MAPPING NEW REGION
16         ;
17         ; Reset all mapping for job
18         ;
19 000204 005002              CLR    R2
20 000206 005062 0000000 11$:  CLR    RPAR(R2)      ; CLEAR PAR
21 000212 005062 0000000      CLR    RPDR(R2)      ; AND PDR
22 000216 062702 0000002      ADD    #2,R2
23 000222 020227 0000016      CMP    R2,#14.        ; DONE ALL?
24 000226 101767              BLOS   11$            ; BR IF MORE TO DO
25 000230 004767 0000000      CALL  SETMAP         ; RESET MAPPING FOR JOB
26 000234 000167 0000000      JMP    EMTXIT        ; FINISHED
27         ;
28         ; Map new virtual address to physical region
29         ;
30 000240 016702 0000020 10$:  MOV    EMTBLK+2,R2    ; GET PAR #
31 000244 020227 0000007      CMP    R2,#7         ; MUST BE IN RANGE 0-7
32 000250 101404              BLOS   1$            ; BR IF OK
33 000252 012700 177767      MOV    #-11,R0       ; ABORT IF TOO LARGE
34 000256 000167 0000000      JMP    SETERR
35 000262 006302 1$:      ASL    R2              ; CONVERT TO WORD TABLE INDEX
36 000264 016704 0000040      MOV    EMTBLK+4,R4   ; GET PHYSICAL ADDRESS BASE
37         ;
38         ; Set up mapping for each PAR that is affected by this request.
39         ; R2 = 2*PAR #
40         ; R4 = Physical address base
41         ; R5 = Size of region being mapped
42         ;
43 000270 010500 2$:      MOV    R5,R0          ; GET # 64-BYTE BLOCKS LEFT TO MAP
44 000272 001450              BEQ    5$            ; BR IF FINISHED
45 000274 020027 000200      CMP    R0,#200       ; EACH PAR CAN ONLY MAP 200 BLOCKS
46 000300 101402              BLOS   3$            ; BR IF OK
47 000302 012700 000200      MOV    #200,R0       ; MAP 200 BLOCKS THROUGH THIS PAR
48 000306 010462 0000000 3$:      MOV    R4,RPAR(R2)    ; SET PAR BASE ADDRESS
49 000312 010462 0000000      MOV    R4,UPARO(R2)  ; LOAD HARDWARE MAP REGISTER
50 000316 010462 0000000      MOV    R4,CUPARO(R2) ; SAVE MAPPING INFO IN CONTEXT BLOCK
51 000322 060004              ADD    R0,R4         ; ADVANCE PAR BASE ADDRESS
52 000324 160005              SUB    R0,R5         ; DECREASE # BLOCKS REMAINING TO BE MAPPED
53 000326 005300              DEC    R0            ; PDR VALUE IS ACTUAL-1
54 000330 000300              SWAB   R0            ; POSITION SIZE FOR PLF FIELD IN PDR
55 000332 042700 100377      BIC    #^C<77400>,R0 ; CLEAR ALL BUT PLF FIELD IN PDR VALUE
56 000336 052700 0000002      BIS    #2,R0         ; SET READ-ENABLE FLAG FOR PDR
57 000342 105767 0000100      TSTB  EMTBLK+10     ; IS WRITE ACCESS ALLOWED?

```

```
58 000346 001402          BEQ      4$          ;BR IF NOT
59 000350 052700 000004    BIS      #4,R0        ;SET WRITE-ENABLE FLAG FOR PDR
60 000354 105767 0000110   4$:    TSTB     EMTBLK+11 ;BYPASS CACHE?
61 000360 001402          BEQ      6$          ;BR IF NOT
62 000362 052700 100000    BIS      #100000,R0   ;SET CACHE-BYPASS FLAG
63 000366 010062 0000000   6$:    MOV      R0,RPDR(R2) ;SET THE PDR VALUE
64 000372 010062 0000000    MOV      R0,UPDR0(R2) ;LOAD HARDWARE MAP REGISTER
65 000376 010062 0000000    MOV      R0,CUPDR0(R2) ;SAVE MAPPING INFORMATION IN JOB CONTEXT BLOCK
66 000402 062702 000002    ADD      #2,R2        ;ADVANCE PAR INDEX #
67 000406 020227 000016    CMP      R2,#2*7     ;ONLY DO UP TO PAR # 7
68 000412 101726          BLOS     2$          ;LOOP IF MORE PAR'S TO MAP
69                          ;
70                          ; Finished
71                          ;
72 000414 000167 0000000   5$:    JMP      EMTXIT    ;FINISHED
```

RTFREZ -- Lock job in memory without repositioning

```

1          .SBTTL  RTFREZ -- Lock job in memory without repositioning
2          ;-----
3          ; The RTFREZ emt locked a job in memory without repositioning it.
4          ;
5 000420  032767  0000000 0000000 RTFREZ: BIT    #P0$LOK,PRIVCO ;Do we have privilege to do this?
6 000426  001001                BNE     20$      ;Br if yes
7 000430  005000                CLR     RO        ;Return error code 0
8 000432  052761  0000000 0000000 20$:  BIS     #$MLOCK,LSW6(R1);LOCK JOB IN MEMORY
9 000440  000167  0000000                JMP     EMTXIT    ;FINISHED

```

```

10
11         .SBTTL  RTUNLK -- Unlock job from memory
12         ;-----
13         ; The RTUNLK emt unlockes a job from memory.
14         ;
15 000444  042761  0000000 0000000 RTUNLK: BIC     #$MLOCK,LSW6(R1);UNLOCK FROM MEMORY
16 000452  000167  0000000                JMP     EMTXIT    ;FINISHED

```

RTPEEK -- Peek at I/O page

```

1          .SBTTL  RTPEEK -- Peek at I/O page
2          ;-----
3          ; RTPEEK emt is used to access a single word in the I/O page.
4          ;
5          ; Inputs:
6          ;   Arg1 = Address of cell in I/O page to be accessed.
7          ;
8          ; Outputs:
9          ;   Contents of cell are returned to user in R0.
10         ;
11 000456 004767 001250 RTPEEK: CALL  PRVMEM          ;Can we access physical memory?
12 000462 017767 0000020 000000G      MOV    @EMTBLK+2,UR0  ;PEEK AT I/O PAGE
13 000470 000421          BR      RTXIT

```

```

14         .SBTTL  RTPOKE -- Poke into I/O page
15         ;-----
16         ; RTPOKE emt is used to store a word into a cell in the I/O page.
17         ;
18         ; Inputs:
19         ;   Arg1 = Address of cell in I/O page to be accessed.
20         ;   Arg2 = Data value to be stored into I/O page cell.
21         ;
22         ;
23 000472 004767 001234 RTPOKE: CALL  PRVMEM          ;Can we access physical memory?
24 000476 016777 0000040 000002G      MOV    EMTBLK+4,@EMTBLK+2;STORE INTO CELL IN I/O PAGE
25 000504 000413          BR      RTXIT

```

```

26         .SBTTL  RTBIS  -- Bit set into I/O page
27         ;-----
28         ; RTBIS emt is used to do a bit set into a word in the I/O page.
29         ;
30         ; Inputs:
31         ;   Arg1 = Address of cell in I/O page to be accessed.
32         ;   Arg2 = Data value to be ORed into I/O cell.
33         ;
34         ;
35 000506 004767 001220 RTBIS:  CALL  PRVMEM          ;Can we access physical memory?
36 000512 056777 0000040 000002G      BIS    EMTBLK+4,@EMTBLK+2;DO BIS INTO I/O PAGE CELL
37 000520 000405          BR      RTXIT

```

```

38         .SBTTL  RTBIC  -- Bit clear into I/O page
39         ;-----
40         ; RTBIC emt is used to do a bit clear into a word in the I/O page.
41         ;
42         ; Inputs:
43         ;   Arg1 = Address of cell in I/O page to be accessed.
44         ;   Arg2 = Data value to be used as a bit clear mask.
45         ;
46         ;
47 000522 004767 001204 RTBIC: CALL  PRVMEM          ;Can we access physical memory?
48 000526 046777 0000040 000002G      BIC    EMTBLK+4,@EMTBLK+2;DO BIC INTO I/O PAGE CELL
49 000534 000167 000000G      RTXIT: JMP    EMTXIT          ;FINISHED WITH EMT

```

RTIOMP -- Map par7 to I/O page

```
1 .SBTTL RTIOMP -- Map par7 to I/O page
2 ;-----
3 ; RTIOMP is called to map the upper 8Kb (par7) of the job's virtual
4 ; address space to the I/O page.
5 ;
6 000540 004767 001166 RTIOMP: CALL PRVMEM ;Can we access I/O page?
7 000544 052761 000000G 000000G BIS #IOMAP,LSW6(R1);SET FLAG SAYING WE ARE ACCESSING I/O PAGE
8 000552 004767 000000G RTMAP: CALL SETMAP ;RELOAD MAPPING REGISTERS FOR JOB
9 000556 000766 BR RTXIT
10
11 .SBTTL RTRMMP -- Map par7 to simulated Rmon
12 ;-----
13 ; RTRMMP emt is used to map the par7 region of the user's job back to
14 ; the simulated Rmon.
15 ;
16 000560 042761 000000G 000000G RTRMMP: BIC #IOMAP,LSW6(R1);SAY WE ARE NO LONGER MAPPED TO I/O PAGE
17 000566 000771 BR RTMAP ;RELOAD MAPPING REGISTERS FOR JOB
```

.SPND -- Suspend job's execution

```

1          .SBTTL .SPND -- Suspend job's execution
2          ;-----
3          ; The .SPND emt is used to suspend the job's execution until a .RSUM
4          ; is done.
5          ;
6 000570 005361 0000000 RTSPND: DEC      LSPND(R1)      ;DEC SUSPENSION COUNTER FOR JOB
7 000574 002023          BGE      1$          ;BR IF WE SHOULD NOT SUSPEND JOB
8          ; Don't suspend job if running in a completion routine.
9 000576 105767 0000000          TSTB     CURCP      ;ARE WE RUNNING IN A COMPLETION ROUTINE NOW?
10 000602 001020          BNE      1$          ;BR IF YES
11         ;
12         ; Suspend execution of the job until a .RSUM emt is done.
13         ;
14 000604          3$:  DISABL          ;;;DISABLE INTERRUPTS
15 000612 005761 0000000          TST      LSPND(R1)      ;;;HAS RESUME BEEN ISSUED?
16 000616 002007          BGE      4$          ;;;BR IF WE SHOULD RESTART JOB
17 000620 012700 0000000          MOV      #S$SPND,RO      ;;;JOB SUSPENDED STATE
18 000624 004767 0000000          CALL     QHDSPN          ;;;SUSPEND THE JOB ** ENABLES INTERRUPTS **
19 000630 004767 0000000          CALL     CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
20 000634 000763          BR       3$          ;ATTEMPT TO RESUME THE JOB
21         ;
22         ; Finished doing the .spnd -- resume the job's execution.
23         ;
24 000636          4$:  ENABL          ;;;ENABLE INTERRUPTS
25 000644 000167 0000000          1$:  JMP      EMTXIT          ;FINISHED WITH .SPND
26         ;
27         .SBTTL .RSUM -- Resume job's execution
28         ;-----
29         ; The .RSUM emt is used to resume execution of a job that is asleep
30         ; because of doing a .SPND.
31         ;
32 000650 005261 0000000 RTRSUM: INC      LSPND(R1)      ;INC JOB'S .SPND COUNTER
33 000654 000167 0000000          JMP      EMTXIT          ;THAT'S ALL WE HAVE TO DO

```

RTCMPL -- Schedule a completion routine

```

1          .SBTTL  RTCMPL -- Schedule a completion routine
2          ;-----
3          ; The RTCMPL EMT is used to schedule a completion routine for the job.
4          ;
5          ; The form of the EMT is
6          ;
7          ;     ENT      375
8          ;
9          ; With R0 pointing to the following EMT argument block:
10         ;
11         ;     .BYTE   21,140
12         ;     .WORD   completion_routine_address
13         ;     .WORD   completion_routine_priority
14         ;     .WORD   value_to_pass_in_R1
15         ;     .WORD   0
16         ;
17 000660  RTCMPL:
18         ;
19         ; Get a completion routine queue element
20         ;
21 000660  004767  000000G      CALL      GETQ          ;Get a free completion queue element
22         ;
23         ; Set up the queue element
24         ;
25 000664  010104              MOV      R1,R4          ;Carry queue element pointer in R4
26 000666  116701  000000G      MOVVB   CORUSR,R1      ;Get job index number
27 000672  110164  000000G      MOVVB   R1,CQ$JOB(R4) ;Set job number in queue element
28 000676  012703  000002G      MOV      #EMTBLK+2,R3 ;Get pointer to block with EMT arguments
29 000702  012364  000000G      MOV      (R3)+,CQ$RTN(R4);Set address of completion routine
30 000706  012302              MOV      (R3)+,R2      ;Get execution priority value
31 000710  001012              BNE     2$           ;Br if priority not = 0
32 000712  112764  000000G  000000G  MOVVB   #S$TWFN,CQ$RNS(R4);Set non-real-time execution state
33 000720  116164  000000G  000000G  MOVVB   LBPRI(R1),CQ$PRI(R4);Set execution priority
34 000726  112764  000000G  000000G  MOVVB   #CP$STD,CQ$CP(R4);Set standard compl rtn class priority
35 000734  000420              BR      4$           ;
36 000736  112764  000000G  000000G  2$:  MOVVB   #CP$RT,CQ$CP(R4);Set real-time completion class priority
37 000744  112764  000000G  000000G  MOVVB   #S$RT,CQ$RNS(R4);Set real-time execution state
38 000752  116700  000000G      MOVVB   VPRIHI,R0     ;Get base real-time priority
39 000756  060002              ADD     R0,R2        ;Add base real-time priority
40 000760  020227  000000G      CMP     R2,#MAXPRI   ;Make sure we don't overflow
41 000764  101402              BLOS   3$           ;Br if ok
42 000766  012702  000000G      MOV     #MAXPRI,R2   ;Truncate to max allowed
43 000772  110264  000000G      3$:  MOVVB   R2,CQ$PRI(R4) ;Set execution priority value
44 000776  012364  000000G      4$:  MOV     (R3)+,CQ$R1(R4);Set value to pass to completion routine in R1
45 001002  005064  000000G      CLR    CQ$R0(R4)    ;Pass 0 in R0
46 001006  013764  000000G  000000G  MOV     @#KPAR5,CQ$PA5(R4);Save current kernel PAR 5 mapping
47         ;
48         ; Queue the completion routine request
49         ;
50 001014  004767  000000G      CALL    QCOMPL       ;Queue the completion routine
51         ;
52         ; Finished
53         ;
54 001020  000167  000000G      JMP     EMTXIT       ;Finished with EMT

```

```
1          .SBTTL  RTEXJB -- Gain exclusive use of system
2          ;-----
3          ; The RTEXJB EMT is used to obtain exclusive use of the system
4          ; by a real-time job. All other jobs are suspended until the RTALJB
5          ; EMT is used to release exclusive access.
6          ;
7 001024 004767 000662 RTEXJB: CALL  PRVRT          ;Do we have Real-time privilege?
8          ;
9          ; Set flag that says this job has exclusive access to the system
10         ;
11 001030 110167 0000000 MOVB  R1,EXCJOB          ;SAY THIS JOB HAS EXCLUSIVE ACCESS
12         ;
13         ; Finished
14         ;
15 001034 000167 0000000 JMP  EMTXIT          ;EXIT FROM EMT
16         ;
17         .SBTTL  RTALJB -- Release exclusive access claim to system
18         ;-----
19         ; The RTALJB emt is used to release exclusive access to the system.
20         ;
21 001040 105067 0000000 RTALJB: CLRB  EXCJOB          ;SAY WE NO LONGER HAVE EXCLUSIVE ACCESS
22         ;
23         ; Set flag to cause execution scheduler to be called
24         ;
25 001044 105267 0000000 INCB  DOSCHD          ;CALL THE SCHEDULER LATER
26         ;
27         ; Finished
28         ;
29 001050 000167 0000000 JMP  EMTXIT          ;EXIT FROM EMT
```

RTSPL -- Set processor priority level

```

1          .SBTTL RTSPL -- Set processor priority level
2          ;-----
3          ; The RTSPL EMT is used to set the processor priority level that
4          ; will be used while running in user mode.
5          ;
6          ; Inputs:
7          ; Arg1 = Processor priority level (0 to 7).
8          ;
9 001054 004767 000632 RTSPL: CALL PRVRT ; Do we have real-time privilege?
10 001060 016702 0000020 MOV EMTBLK+2,R2 ; GET REQUESTED PROCESSOR PRIORITY LEVEL
11 001064 042702 177770 BIC #177770,R2 ; CLEAR ALL BUT PRIORITY LEVEL FIELD
12 001070 072227 000005 ASH #5,R2 ; POSITION TO PRIORITY FIELD IN PS
13 001074 042767 000340 0000000 BIC #340,EMTPS ; CLEAR OLD PRIORITY FIELD IN PS
14 001102 050267 0000000 BIS R2,EMTPS ; STORE NEW PRIORITY LEVEL INTO PS
15 001106 000167 0000000 JMP EMTXIT ; DO EMT EXIT WHICH WILL LOAD PS

```

RTCVEC --- Connect completion routine to interrupt vector

```

1          .SBTTL  RTCVEC --- Connect completion routine to interrupt vector
2          ;-----
3          ; RTCVEC emt is used to connect an interrupt vector to a user completion
4          ; routine.
5          ;
6          ; Inputs:
7          ;   Arg1 = Address of interrupt vector.
8          ;   Arg2 = Address of completion routine.
9          ;   Arg3 = Priority of completion routine (0 to 7).
10         ;
11         ; Entry point for directly connected interrupt service routine
12         ;
13 001112 004767 000574 RTDVEC: CALL  PRVRT          ; Do we have real-time privilege?
14 001116 032761 0000000 0000000 BIT    ##MLOCK,LSW6(R1); IS JOB LOCKED IN MEMORY?
15 001124 001004          BNE    1$          ; BR IF YES
16 001126 012700 000003  MOV    #3,R0          ; RETURN EMT ERROR CODE 3 IF NOT
17 001132 000167 0000000  JMP    SETERR
18 001136 012705 0000000 1$:  MOV    #VF$DIR,R5      ; SET DIRECTLY-CONNECTED INTERRUPT FLAG
19 001142 000401          BR     RTXVEC          ; ENTER COMMON CODE
20         ;
21         ; Entry point for completion routine type service routine
22         ;
23 001144 005005 RTDVEC: CLR    R5          ; CLEAR DIRECTLY-CONNECTED INTERRUPT FLAG
24         ;
25         ; See if interrupt vector is already connected to a service routine
26         ;
27 001146 016700 0000020 RTXVEC: MOV    EMTBLK+2,R0      ; GET ADDRESS OF VECTOR WE ARE CONNECTING TO
28 001152 006200          ASR    R0          ; DROP LOW-ORDER BIT
29 001154 010003          MOV    R0,R3          ; SAVE VECTOR VALUE
30 001156 004767 000570  CALL   SRCVEC          ; SEE IF THERE IS A VECTOR CONTROL BLOCK ALREADY
31 001162 103407          BCS   1$          ; BR IF NO CONTROL BLOCK FOR THIS VECTOR
32 001164 126201 0000000  CMPB  VC$JOB(R2),R1     ; IS OUR JOB THE ONE THAT OWNS THE VECTOR?
33 001170 001414          BEQ   2$          ; BR IF YES
34 001172 012700 000002  MOV    #2,R0          ; ERROR CODE 2 MEANS SOMEONE ELSE OWNS VECTOR
35 001176 000167 0000000  JMP    SETERR
36         ;
37         ; Vector is free. Get a vector control block and set it up.
38         ;
39 001202 005000 1$:  CLR    R0          ; GET A FREE VECTOR CONTROL BLOCK
40 001204 004767 000542  CALL   SRCVEC
41 001210 103004          BCC   2$          ; BR IF GOT ONE
42 001212 012700 000001  MOV    #1,R0          ; NO FREE VECTOR CONTROL BLOCKS
43 001216 000167 0000000  JMP    SETERR
44         ;
45         ; Found a free vector control block (Address is in R2).
46         ; Set it up.
47         ;
48 001222 110162 0000000 2$:  MOVB  R1,VC$JOB(R2)   ; SET JOB NUMBER
49 001226 110362 0000000  MOVB  R3,VC$VEC(R2)   ; SET ADDRESS OF VECTOR
50 001232 110562 0000000  MOVB  R5,VC$FLG(R2)   ; SET CONTROL FLAGS (VF$xxx)
51 001236 016762 0000040 0000000  MOV    EMTBLK+4,VC$RTN(R2); SET ADDRESS OF COMPLETION ROUTINE
52 001244 016700 0000060  MOV    EMTBLK+6,R0     ; GET COMPLETION ROUTINE PRIORITY
53 001250 042700 177770  BIC    #^C7,R0        ; FORCE TO RANGE 0-7
54 001254 110062 0000000  MOVB  R0,VC$PRI(R2)   ; SAVE PRIORITY IN VECTOR CONTROL BLOCK
55         ;
56         ; Make vector point to vector control block.
57         ;

```

RTCVEC -- Connect completion routine to interrupt vector

```
58 001260 006303          ASL   R3           ;GET ADDRESS OF VECTOR
59 001262 010223          MOV   R2,(R3)+      ;SET VECTOR PC -- SEND INT TO INT CONTROL BLOCK
60 001264 012713 000340   MOV   #340,(R3)     ;SET VECTOR PS
61                               ;
62                               ; Finished
63                               ;
64 001270 000167 0000000  JMP   EMTXIT
```

```

1          .SBTTL  RTRVEC -- Release interrupt vector
2          ;-----
3          ; RTRVEC emt is used to release an interrupt vector connection.
4          ;
5          ; Inputs:
6          ; Arg1 = Address of vector.
7          ;
8 001274 004767 000412 RTRVEC: CALL  PRVRT          ;Do we have real-time privilege?
9 001300 016700 0000020 MOV    EMTBLK+2,R0    ;GET ADDRESS OF VECTOR
10 001304 006200      ASR    R0              ;DROP LOW-ORDER BIT
11 001306 004767 000440 CALL  SRCVEC         ;SEARCH FOR VECTOR CONTROL BLOCK
12 001312 103405      BCS    1$              ;BR IF NONE FOR THIS VECTOR
13 001314 120162 0000000 CMPB  R1,VC$JOB(R2)  ;DO WE OWN THE VECTOR?
14 001320 001002      BNE    1$              ;BR IF NOT
15 001322 004767 0000004 CALL  RELVEC        ;RELEASE THE VECTOR
16 001326 000167 0000000 1$:  JMP    EMTXIT
17          ;-----
18          ; RELVEC is called to release a user's link with an interrupt vector.
19          ; The vector is set to point to the unexpected interrupt routine and
20          ; the vector control block is freed.
21          ;
22          ; Inputs:
23          ; R2 = Address of vector control block.
24          ;
25          ;
26 001332 010346 RELVEC: MOV    R3,-(SP)
27          ;
28          ; If .DEVICE list already specified a reset address for this vector,
29          ; don't make it point to normal unexpected-interrupt location.
30          ;
31 001334 132762 0000000 0000000 BITB  #VF$DET,VC$FLG(R2);Did .DEVICE disconnect the interrupt?
32 001342 001031      BNE    2$              ;Br if yes
33          ;
34          ; Set interrupt vector to point to unexpected interrupt routine.
35          ;
36 001344 005003      CLR    R3              ;GET ADDRESS OF INT VECTOR FROM VECTOR CONTROL
37 001346 156203 0000000 BISB  VC$VEC(R2),R3
38 001352 006303      ASL    R3              ;CONVERT TO REAL ADDRESS
39 001354 010300      MOV    R3,R0
40 001356 105767 0000000 TSTB  VUXIFL        ;SHOULD WE CRASH OR IGNORE UNEXPECTED INTS?
41 001362 001005      BNE    1$              ;BR IF WE SHOULD CRASH
42 001364 012720 0000000 MOV    #UEXRTN,(R0)+ ;SET PC TO ROUTINE TO IGNORE INTS
43 001370 012710 000340 MOV    #340,(R0)    ;SET PS PRIO=7 FOR INTERRUPT
44 001374 000414      BR     2$
45 001376 012720 0000000 1$:  MOV    #UEXINT,(R0)+ ;SET VECTOR TO GO TO UNEXPECTED INT ROUTINE
46          ;
47          ; Set PS in interrupt vector to encode interrupt address.
48          ;
49 001402 072327 177776 ASH    #-2,R3        ;DROP LOW-ORDER 2 BITS OF VECTOR ADDRESS
50 001406 010346      MOV    R3,-(SP)
51 001410 042716 177760 BIC    #^C17,(SP)   ;CLEAR ALL BUT N-Z-V-C FIELDS
52 001414 006303      ASL    R3              ;SHIFT OVER ADDRESS TO AVOID T FLAG
53 001416 042703 177437 BIC    #^C340,R3    ;CLEAR ALL BUT PRIORITY FIELD
54 001422 052603      BIS    (SP)+,R3     ;COMBINE N-Z-V-C WITH PRIO FIELD
55 001424 010310      MOV    R3,(R0)     ;STORE INTO VECTOR PS WORD
56          ;
57          ; Free the vector control block

```

```
58 ;  
59 001426 105062 0000000 2#: CLRB VC#JOB(R2)  
60 001432 105062 0000000 CLRB VC#VEC(R2)  
61 ;  
62 ; Finished  
63 ;  
64 001436 012603 MOV (SP)+, R3  
65 001440 000207 RETURN
```

```
1          .SBTTL  RTDEV  -- .DEVICE Emt
2          ;-----
3          ; The .DEVICE emt is used to declare a list of addresses and values to
4          ; be stored when a job exits.
5          ;
6 001442  004767  000244  RTDEV:  CALL  PRVRT          ;Do we have real-time privilege?
7          ;
8          ; Determine if this is a simple or linked type list
9          ;
10 001446  105767  0000000  1#:    TSTB   EMTBLK          ;SIMPLE OR LINKED LIST?
11 001452  001004          BNE     2#          ;BR IF LINKED LIST
12          ;
13          ; Simple (non-linked) list
14          ;
15 001454  016767  0000020 0000000  MOV     EMTBLK+2,DEVLS ;SET ADDRESS OF .DEVICE LIST
16 001462  000407          BR      9#          ;FINISHED
17          ;
18          ; Linked list
19          ;
20 001464  016746  0000000  2#:    MOV     DEVL, -(SP) ;PUSH ADDRESS OF CURRENT LIST HEAD
21 001470  106677  0000020          MTPD    @EMTBLK+2 ;STORE AS FORWARD LINK FROM NEW LIST
22 001474  016767  0000020 0000000  MOV     EMTBLK+2,DEVL ;SAVE ADDRESS OF NEW LIST
23          ;
24          ; Finished
25          ;
26 001502  000167  0000000  9#:    JMP     EMTXIT
```

RTSTOP -- Real-time cleanup at end of job execution

```

1          .SBTTL  RTSTOP -- Real-time cleanup at end of job execution
2          ;-----
3          ; RTSTOP is called when a job exits for any reason.
4          ; It does any necessary real-time cleanup for the job.
5          ; This consists of the following things:
6          ; 1. Process any specified .DEVICE list.
7          ; 2. Disconnect any interrupt vectors attached to the job.
8          ; 3. Release exclusive access of system if job has gotten it.
9          ;
10         ; Inputs:
11         ; R1 = Current job number
12         ;
13 001506 010246 RTSTOP: MOV      R2,-(SP)
14         ;
15         ; If job did a .DEVICE emt, process the device reset list.
16         ;
17         ; Process linked type lists first
18 001510 016702 0000000 1$:  MOV      DEVLL,R2      ; IS THERE A LINKED TYPE LIST TO PROCESS?
19 001514 001415          BEQ      3$              ; BR IF NOT
20 001516 005067 0000000  CLR      DEVLL          ; CLEAR IN CASE WE TRAP
21 001522 106522          MFPD     (R2)+          ; GET ADDRESS OF FORWARD LINK FROM LIST HEAD
22 001524 012667 0000000  MOV      (SP)+,DEVLL    ; SAVE THIS ADDRESS AS NEW HEAD OF LINKED LIST
23 001530 106522          2$:  MFPD     (R2)+          ; GET ADDRESS TO STORE INTO
24 001532 012600          MOV      (SP)+,R0
25 001534 001765          BEQ      1$              ; BR IF REACHED END OF LIST
26 001536 106522          MFPD     (R2)+          ; GET VALUE TO STORE INTO CELL
27 001540 012610          MOV      (SP)+,(R0)      ; DO THE STORE
28 001542 004767 000112  CALL     CKVREL        ; See if we just released an interrupt conctn.
29 001546 000770          BR       2$              ; GO PROCESS NEXT ITEM IN LIST
30         ;
31         ; Now process non-linked type lists
32 001550 016702 0000000 3$:  MOV      DEVLS,R2      ; ANY NON-LINKED .DEVICE LIST?
33 001554 001412          BEQ      INTSTP         ; BR IF NOT
34 001556 005067 0000000  CLR      DEVLS          ; CLEAR IN CASE WE TRAP
35 001562 106522          4$:  MFPD     (R2)+          ; GET ADDRESS TO STORE INTO
36 001564 012600          MOV      (SP)+,R0
37 001566 001405          BEQ      INTSTP         ; BR IF REACHED END OF LIST
38 001570 106522          MFPD     (R2)+          ; GET VALUE TO STORE
39 001572 012610          MOV      (SP)+,(R0)      ; DO THE STORE
40 001574 004767 000060  CALL     CKVREL        ; See if we released an interrupt connection
41         ;
42         ; Disconnect all interrupt vectors associated with this job
43         ;
44 001602 012702 0000000 INTSTP: MOV      #VCBBAS,R2  ; POINT TO 1ST VECTOR CONTROL BLOCK
45 001606 020227 0000000 1$:  CMP      R2,#VCBEND    ; CHECKED ALL VECTOR CONTROL BLOCKS?
46 001612 103010          BHI     4$              ; BR IF YES
47 001614 120162 0000000  CMPB     R1,VC$JOB(R2)   ; IS THIS VCB IN USE BY THIS JOB?
48 001620 001002          BNE     2$              ; BR IF NOT
49 001622 004767 177504  CALL     RELVEC         ; RELEASE THE VECTOR AND THE VCB
50 001626 062702 0000000  2$:  ADD      #VC#$SZ,R2    ; POINT TO NEXT VCB
51 001632 000765          BR       1$              ;
52         ;
53         ; If job has exclusive access to the system, release it
54         ;
55 001634 120167 0000000  4$:  CMPB     R1,EXCJOB    ; DOES THIS JOB HAVE EXCLUSIVE ACCESS TO SYS?
56 001640 001005          BNE     3$              ; BR IF NOT
57 001642 105767 0000000  TSTB     CINFLG        ; IS THIS JOB DOING A CHAIN NOW?

```

```
58 001646 001002          BNE      3#          ; IF YES THEN RETAIN EXCLUSIVE ACCESS
59 001650 105067 0000000  CLRB    EXCJOB        ; RELEASE EXCLUSIVE ACCESS
60                                     ;
61                                     ; Finished
62                                     ;
63 001654 012602          3#:    MOV      (SP)+, R2
64 001656 000207          RETURN
```

RTSTOP -- Real-time cleanup at end of job execution

```

1      ; -----
2      ; See if the cell being stored into by the .DEVICE list is an interrupt
3      ; vector.  If so, mark the vector control block so that we don't alter
4      ; the interrupt location later.
5      ;
6      ; Inputs:
7      ; RO = Address of cell being modified by .DEVICE list processing.
8      ;
9 001660 010246 CKVREL: MOV     R2,-(SP)
10     ;
11     ; See if location being modified is an interrupt vector
12     ;
13 001662 020027 000500      CMP     RO,#500      ;Could this be an interrupt vector?
14 001666 103007      BHIS    9$          ;Br if not
15 001670 006200      ASR     RO          ;Get vector address / 2
16 001672 004767 000054      CALL   SRCVEC       ;Try to find vector control block
17 001676 103403      BCS     9$          ;Br if no vector attachment
18     ;
19     ; Set flag in vector control block which prevents us from connecting
20     ; this vector to unexpected-interrupt location later.
21     ;
22 001700 152762 0000000 0000000  BISB   #VF$DET,VC$FLG(R2) ;Say vector has been disconnected
23     ;
24     ; Finished
25     ;
26 001706 012602      9$:    MOV     (SP)+,R2
27 001710 000207      RETURN

```

** Subroutines **

```

1          .SBTTL  ** Subroutines **
2          .SBTTL  PRVRT  -- Determine if job has real-time privilege
3          ;-----
4          ; Determine if the job has real-time privilege.
5          ; If not, error code 0 is returned for the EMT.
6          ;
7 001712  032767  000000G 000000G PRVRT:  BIT    #P0$RT,PRIVCO  ;Does job have real-time privilege?
8 001720  001003                BNE    9$                ;Br if yes
9 001722  005000                CLR    R0                ;Return error code 0
10 001724  000167  000000G      JMP    SETERR
11 001730  000207                9$:   RETURN
12
13          .SBTTL  PRVMEM -- Check for access authorization to phys memory
14          ;-----
15          ; Determine if the job can access physical memory.
16          ; If not, error code 0 is returned for the EMT.
17          ;
18 001732  032767  000000G 000000G PRVMEM: BIT    #P0$MEM,PRIVCO  ;Can we access physical memory?
19 001740  001003                BNE    9$                ;Br if yes
20 001742  005000                CLR    R0                ;Return error code 0
21 001744  000167  000000G      JMP    SETERR
22 001750  000207                9$:   RETURN

```

SRCVEC -- Search for vector control block

```

1          .SBTTL  SRCVEC -- Search for vector control block
2          ;-----
3          ; SRCVEC is called to search for a vector control block associated
4          ; with a particular interrupt vector.
5          ;
6          ; Inputs:
7          ;   R0 = Address of vector / 2
8          ;
9          ; Outputs:
10         ;   C-flag set on return if no vector control block found for vector.
11         ;   R2 = Address of vector control block if one found.
12         ;
13 001752  012702  0000000  SRCVEC:  MOV    #VCBBAS,R2    ;GET ADDRESS OF BASE OF CONTROL BLOCK AREA
14 001756  020227  0000000  1$:    CMP    R2,#VCBEND    ;CHECKED ALL BLOCKS?
15 001762  103006                BHS    4$                ;BR IF YES
16 001764  120062  0000000  CMPB   R0,VC$VEC(R2)    ;IS THIS CONTROL BLOCK FOR THE VECTOR?
17 001770  001405                BEQ    2$                ;BR IF YES
18 001772  062702  0000000  ADD    #VC$$SZ,R2      ;POINT TO NEXT VECTOR CONTROL BLOCK
19 001776  000767                BR     1$
20 002000  000261  4$:    SEC                    ;SIGNAL FAILURE ON RETURN
21 002002  000401                BR     3$
22 002004  000241  2$:    CLC                    ;SIGNAL SUCCESS ON RETURN
23 002006  000207  3$:    RETURN
24         000001          .END

```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 153 Words (1 Pages)
 Size of core pool: 17920 Words (70 Pages)
 Operating system: RT-11

Elapsed time: 00:00:14.47
 DK: TSRTX, LP: TSRTX=DK: TSRTX. MAC/C/N: SYM

RTBIS	2-34	6-35#					
RTCMPL	2-48	9-17#					
RTCVD	2-31	3-15#					
RTCVEC	2-40	12-23#					
RTDEV	1-30	14-6#					
RTDVEC	2-47	12-13#					
RTEMT	1-30	2-17#					
RTEMTV	2-26	2-31#	2-50				
RTEXJB	2-43	10-7#					
RTFREQ	2-42	5-5#					
RTIOMP	2-36	7-6#					
RTLCK	1-37	2-38					
RTMAP	7-8#	7-17					
RTPEEK	2-32	6-11#					
RTPOKE	2-33	6-23#					
RTRMMP	2-37	7-16#					
RTRSUM	1-29	8-32#					
RTRVEC	2-41	13-8#					
RTSPL	2-45	11-9#					
RTSPND	1-29	8-6#					
RTSTOP	1-30	15-13#					
RTUNLK	2-39	5-15#					
RTVMAP	2-46	4-13#					
RTXIT	6-13	6-25	6-37	6-49#	7-9		
RTXVEC	12-19	12-27#					
S#RT	1-48	9-37					
S#SPND	1-41	8-17					
S#TWFN	1-38	9-32					
SETERR	1-34	4-34	12-17	12-35	12-43	17-10	17-21
SETMAP	1-35	4-25	7-8				
SRCVEC	12-30	12-40	13-11	16-16	18-13#		
TSRTX	1-13#						
UEXINT	1-36	13-45					
UEXRTN	1-39	13-42					
UPARO	1-45	4-49*					
UPDRO	1-45	4-64*					
UPMODE	1-46	1-46					
URO	1-37	6-12*					
VC##SZ	1-40	15-50	18-18				
VC#FLG	1-44	12-50*	13-31	16-22*			
VC#JOB	1-35	12-32	12-48*	13-13	13-59*	15-47	
VC#PRI	1-37	12-54*					
VC#RTN	1-35	12-51*					
VC#VEC	1-35	12-49*	13-37	13-60*	18-16		
VCBBAS	1-40	15-44	18-13				
VCBEND	1-40	15-45	18-14				
VF#DET	1-44	13-31	16-22				
VF#DIR	1-44	12-18					
VPRIHI	1-48	9-38					
VUXIFL	1-39	13-40					

DISABL	1-54#	8-14
ENABL	1-58#	8-24
OCALL	1-19#	