TSRTX -- Non-resident Real-Time MACRO V05.05 Wednesday 18-Jan-89 14:16
Table of contents

```
                                        .TITLE  TSRTX -- Non-resident Real-Time Support Module
                                        .ENABL  LC
                                        .DSABL  GBL
                                ;-----------------------------------------------------------------
                                ;  TSRTX is the non-resident portion of the TSX-Plus real-time support
                                ;  code.
                                ;
                                ;  Copyright (c) 1980,1981,1982,1983,1984,1985,1986,1987,1988,1989.
                                ;  S&H Computer Systems, Inc.  Nashville, Tennessee
                                ;  All rights reserved.
                                ;
                                        .CSECT  TSRTX
000000                          TSRTX:  .RAD50  /RTX/           ;Overlay id
000000  071670
                                ;
                                ;  Macro definitions
                                ;
                                ;  Macro to call a routine in another system overlay region.
                                ;
                                        .MACRO  OCALL   ENTADD
                                        .IF     B,ENTADD
                                        .ERROR  ;OCALL without entry address
                                        .ENDC
                                        CALL    OVRHC
                                        .WORD   ENTADD
                                        .ENDM   OCALL
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  RTSPND,RTRSUM
                                        .GLOBL  RTSTOP,RTDEV,RTEMT
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  LSW2,SETERR,EMTBLK,BADEMT,LPARBS,EMTXIT
                                        .GLOBL  $IOMAP,LSW6,SETMAP,VC$JOB,VC$VEC,VC$RTN
                                        .GLOBL  UEXINT,CQ$JOB,CQ$RO,CQ$RTN,QCOMPL
                                        .GLOBL  URO,VC$PRI,EMTPS,INTPRI,RTLOCK
                                        .GLOBL  S$TWFN,LBSPRI,$MLOCK,PO$LOK
                                        .GLOBL  OVRHC,UEXRTN,VUXIFL
                                        .GLOBL  VCBBAS,VCBEND,VC$$SZ,PO$MEM,PO$RT,PRIVCO
                                        .GLOBL  LSPND,CURCP,S$SPND,CHKABT
                                        .GLOBL  QHDSPN,DEVLL,DEVLS,CQ$PRI,CQ$PA5,CQ$RNS
                                        .GLOBL  DOSCHD,EXCJOB,CINFLG
                                        .GLOBL  RPAR,RPDR,RDAR,RDDR,VF$DIR,VC$FLG,VF$DET
                                        .GLOBL  UPARO,UPDRO,CUPARO,CUPDRO
                                        .GLOBL  UPMODE,UPMODE,KPAR5,CQ$CP,CP$STD,CP$RT
                                        .GLOBL  CUPARO,CUPDRO,CQ$R1,$UDSPC,LSW11
                                        .GLOBL  GETQ,CORUSR,MAXPRI,S$RT,VPRIHI
                                ;
                                ;-----------------------------------------------------
                                ;  MACROS TO ENABLE AND DISABLE INTERRUPTS.
                                ;
177776                          PS      =       177776          ;PROCESSOR STATUS WORD
                                        .MACRO  DISABL          ;DISABLE INTERRUPTS
                                        BIS     #340,@#PS
                                        .ENDM   DISABL
                                ;
```

```
      58                                        .MACRO  ENABL           ;ENABLE INTERRUPTS
      59                                        BIC     INTPRI,@#PS
      60                                        .ENDM   ENABL
```

```
                                        .SBTTL  RTEMT  -- Real-time emt entry point
                                ;-----------------------------------------------------------------------
                                ;  RTEMT is jumped to from TSEMT whenever one of the real-time emt's
                                ;  is executed.
                                ;  The general form of a real-time emt is
                                ;
                                ;       .BYTE   sub-function,140
                                ;       .WORD   Arg1    ;(optional)
                                ;       .WORD   Arg2    ;(optional)
                                ;       .WORD   Arg3    ;(optional)
                                ;
                                ;  Where sub-function is a code that indicates which real-time
                                ;  function is to be performed. (See vector at end of this routine)
                                ;
                                ;  See if user is authorized to use real-time emt's.
                                ;
000002                          RTEMT:
                                ;
                                ;  Get sub-function code from arg block and jump to processing routine
                                ;
000002  116705  000000G         1$:     MOVB    EMTBLK,R5       ;GET SUB-FUNCTION CODE
000006  006305                          ASL     R5              ;CONVERT TO WORD TABLE INDEX
000010  020527  000044                  CMP     R5,#MXRTFN      ;IS IT VALID SUB-FUNCTION CODE?
000014  101402                          BLOS    2$              ;BR IF OK
000016  000167  000000G                 JMP     BADEMT          ;INVALID SUB-FUNCTION VALUE
000022  000175  000026'         2$:     JMP     @RTEMTV(R5)     ;JUMP TO PROCESSING ROUTINE FOR EMT
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Define sub-function jump vector.
                                ;
000026  000124'                 RTEMTV: .WORD   RTCVAD          ;00 - Convert virtual address to physical
000030  000516'                         .WORD   RTPEEK          ;01 - Peek at I/O page
000032  000532'                         .WORD   RTPOKE          ;02 - Poke into I/O page
000034  000546'                         .WORD   RTBIS           ;03 - Bit set into I/O page
000036  000562'                         .WORD   RTBIC           ;04 - Bit clear into I/O page
000040  000600'                         .WORD   RTIOMP          ;05 - Map virtual PAR6 to I/O page
000042  000620'                         .WORD   RTRMMP          ;06 - Map virtual PAR6 to simulated RMON
000044  000000G                         .WORD   RTLOCK          ;07 - Lock job in low memory
000046  000504'                         .WORD   RTUNLK          ;10 - Unlock job from memory
000050  001204'                         .WORD   RTCVEC          ;11 - Connect completion routine to interrupt
000052  001334'                         .WORD   RTRVEC          ;12 - Disconnect completion rtn from interrupt
000054  000460'                         .WORD   RTFREZ          ;13 - Lock job in memory without repositioning
000056  001064'                         .WORD   RTEXJB          ;14 - Gain exclusive use of system
000060  001100'                         .WORD   RTALJB          ;15 - Release exclusive use of system
000062  001114'                         .WORD   RTSPL           ;16 - Set processor priority level level
000064  000232'                         .WORD   RTVMAP          ;17 - Map virtual region to physical region
000066  001152'                         .WORD   RTDVEC          ;20 - Connect direct interrupt service routine
000070  000720'                         .WORD   RTCMPL          ;21 - Schedule completion routine
000072  000074'                         .WORD   RTCVDA          ;22 - Convert D-space address to physical
                                ;
000044                          MXRTFN  =       .-RTEMTV-2      ;2 * Highest legal subfunction number
```

```
                                        .SBTTL  RTCVAD, RTCVDA -- Convert virtual to physical address
                                ;-----------------------------------------------------------------------
                                ;  The RTCVAD and RTCVDA emts are used to convert a 16-bit virtual address
                                ;  to a 22-bit physical address.  RTCVAD for I-space and RTCVDA for D-space.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Virtual address
                                ;   Arg2 = Address of 2-word block to receive 22-bit address result
                                ;
                                ;  Outputs:
                                ;   Result buffer in user's area receives 22-bit physical address
                                ;   low-order 16-bits stored in 1st word and high-order 2-bits positioned
                                ;   in bit positions 4-5 are stored in 2nd word.
                                ;
                                ;  Convert specially mapped D-space (22,140)
                                ;
000074  032761  000000G 000000G RTCVDA: BIT     #$UDSPC,LSW11(R1) ;Has user enabled D-space?
000102  001410                         BEQ     RTCVAD          ;If not, use I-space routine
000104  004767  000102                 CALL    RTCVC1          ;Split virtual address to par and offset
000110  005762  000000G                TST     RDDR(R2)        ;Is this region specially mapped?
000114  001415                         BEQ     RTCVC3          ;Br if not, use default mapping
000116  016205  000000G                MOV     RDAR(R2),R5     ;Get physical address for this par
000122  000407                         BR      RTCVC2          ;Br to special mapping adjustment
                                ;
                                ;  Convert specially mapped I-space (0,140)
                                ;
000124  004767  000062          RTCVAD: CALL    RTCVC1          ;Split virtual address to par and offset
000130  005762  000000G                TST     RPDR(R2)        ;IS THIS REGION SPECIALLY MAPPED?
000134  001405                         BEQ     RTCVC3          ;BR IF NOT
000136  016205  000000G                MOV     RPAR(R2),R5     ;GET PHYSICAL ADDRESS FOR THIS PAR
000142  042700  160000          RTCVC2: BIC     #160000,R0      ;Remove PAR # from virtual address
000146  000402                         BR      RTCVC4          ;Go merge back with offset and return
                                ;
                                ;  No special mapping, treat as linear offset from job base
                                ;
000150  016105  000000G         RTCVC3: MOV     LPARBS(R1),R5   ;GET BASE 64-BYTE BLOCK NUMBER OF JOB REGION
                                ;
                                ;  PAR base in R5, merge back with offset and return to user
                                ;
000154  005004                  RTCVC4: CLR     R4              ;CLEAR HIGH-ORDER WORD
000156  073427  000006                 ASHC    #6,R4           ;CONVERT TO 22-BIT PHYSICAL ADDR OF JOB BASE
000162  060005                         ADD     R0,R5           ;ADD VIRTUAL ADDRESS offset
000164  005504                         ADC     R4              ;PROPOGATE CARRY
000166  072427  000004                 ASH     #4,R4           ;PUT 2 HIGH-ORDER BITS IN BIT POSITIONS 4-5
000172  016703  000004G                MOV     EMTBLK+4,R3     ;GET ADDRESS OF BUFFER WHERE RESULT GOES
000176  010546                         MOV     R5,-(SP)        ;STACK LOW-ORDER PART OF RESULT
000200  106623                         MTPD    (R3)+           ;STORE INTO USER'S BUFFER
000202  010446                         MOV     R4,-(SP)        ;STACK HIGH-ORDER PART OF RESULT
000204  106613                         MTPD    (R3)            ;STORE INTO USER'S BUFFER
000206  000167  000000G                JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Common subroutine for virtual to physical address conversions
                                ;  Converts user virtual address (from EMTBLK+2) to PAR index in R2
                                ;
000212  016700  000002G         RTCVC1: MOV     EMTBLK+2,R0     ;GET VIRTUAL ADDRESS
000216  010003                         MOV     R0,R3           ;Copy to low order of R2-R3 pair
000220  005002                         CLR     R2              ;SET UP FOR SHIFT
```

```
     58 000222  073227  000003                ASHC    #3,R2           ;GET 3 HIGH-ORDER BITS OF VIRTUAL ADDRESS
     59 000226  006302                        ASL     R2              ;*2 TO CONVERT TO WORD TABLE INDEX
     60 000230  000207                        RETURN
```

```
                        .SBTTL  RTVMAP -- Map virtual address to physical region
                ;-----------------------------------------------------------------------
                ;  This EMT is used to map a virtual region of memory to a specified
                ;  physical region.
                ;
                ;       .BYTE   17,140
                ;       .WORD   Base PAR # (0-7)
                ;       .WORD   Physical base (64-byte block #)
                ;       .WORD   Region length (# 64-byte blocks)
                ;       .BYTE   Access control (0==>read only, 1==>read/write)
                ;       .BYTE   Cache control (0==>enable caching, 1==>bypass cache)
                ;
000232  004767  001534  RTVMAP: CALL    PRVMEM          ;Can we access physical memory?
000236  016705  0000060         MOV     EMTBLK+6,R5     ;GET SIZE OF REGION
000242  001016                  BNE     10$             ;BR IF MAPPING NEW REGION
                ;
                ;  Reset all mapping for job
                ;
000244  005002                  CLR     R2
000246  005062  0000000 11$:    CLR     RPAR(R2)        ;CLEAR PAR
000252  005062  0000000         CLR     RPDR(R2)        ;AND PDR
000256  062702  000002          ADD     #2,R2
000262  020227  000016          CMP     R2,#14.         ;DONE ALL?
000266  101767                  BLOS    11$             ;BR IF MORE TO DO
000270  004767  0000000         CALL    SETMAP          ;RESET MAPPING FOR JOB
000274  000167  0000000         JMP     EMTXIT          ;FINISHED
                ;
                ;  Map new virtual address to physical region
                ;
000300  016702  0000020 10$:    MOV     EMTBLK+2,R2     ;GET PAR #
000304  020227  000007          CMP     R2,#7           ;MUST BE IN RANGE 0-7
000310  101404                  BLOS    1$              ;BR IF OK
000312  012700  177767          MOV     #-11,R0         ;ABORT IF TOO LARGE
000316  000167  0000000         JMP     SETERR
000322  006302          1$:     ASL     R2              ;CONVERT TO WORD TABLE INDEX
000324  016704  0000040         MOV     EMTBLK+4,R4     ;GET PHYSICAL ADDRESS BASE
                ;
                ;  Set up mapping for each PAR that is affected by this request.
                ;    R2 = 2*PAR #
                ;    R4 = Physical address base
                ;    R5 = Size of region being mapped
                ;
000330  010500          2$:     MOV     R5,R0           ;GET # 64-BYTE BLOCKS LEFT TO MAP
000332  001450                  BEQ     5$              ;BR IF FINISHED
000334  020027  000200          CMP     R0,#200         ;EACH PAR CAN ONLY MAP 200 BLOCKS
000340  101402                  BLOS    3$              ;BR IF OK
000342  012700  000200          MOV     #200,R0         ;MAP 200 BLOCKS THROUGH THIS PAR
000346  010462  0000000 3$:     MOV     R4,RPAR(R2)     ;SET PAR BASE ADDRESS
000352  010462  0000000         MOV     R4,UPAR0(R2)    ;LOAD HARDWARE MAP REGISTER
000356  010462  0000000         MOV     R4,CUPAR0(R2)   ;SAVE MAPPING INFO IN CONTEXT BLOCK
000362  060004                  ADD     R0,R4           ;ADVANCE PAR BASE ADDRESS
000364  160005                  SUB     R0,R5           ;DECREASE # BLOCKS REMAINING TO BE MAPPED
000366  005300                  DEC     R0              ;PDR VALUE IS ACTUAL-1
000370  000300                  SWAB    R0              ;POSITION SIZE FOR PLF FIELD IN PDR
000372  042700  100377          BIC     #^C<77400>,R0   ;CLEAR ALL BUT PLF FIELD IN PDR VALUE
000376  052700  000002          BIS     #2,R0           ;SET READ-ENABLE FLAG FOR PDR
000402  105767  0000100         TSTB    EMTBLK+10       ;IS WRITE ACCESS ALLOWED?
```

```
     58 000406  001402                          BEQ     4$              ;BR IF NOT
     59 000410  052700  000004                  BIS     #4,R0           ;SET WRITE-ENABLE FLAG FOR PDR
     60 000414  105767  000011G       4$:      TSTB    EMTBLK+11       ;BYPASS CACHE?
     61 000420  001402                          BEQ     6$              ;BR IF NOT
     62 000422  052700  100000                  BIS     #100000,R0      ;SET CACHE-BYPASS FLAG
     63 000426  010062  000000G       6$:      MOV     R0,RPDR(R2)     ;SET THE PDR VALUE
     64 000432  010062  000000G                 MOV     R0,UPDR0(R2)    ;LOAD HARDWARE MAP REGISTER
     65 000436  010062  000000G                 MOV     R0,CUPDR0(R2)   ;SAVE MAPPING INFORMATION IN JOB CONTEXT BLOCK
     66 000442  062702  000002                  ADD     #2,R2           ;ADVANCE PAR INDEX #
     67 000446  020227  000016                  CMP     R2,#2*7         ;ONLY DO UP TO PAR # 7
     68 000452  101726                          BLOS    2$              ;LOOP IF MORE PAR'S TO MAP
     69                                 ;
     70                                 ;  Finished
     71                                 ;
     72 000454  000167  000000G         5$:     JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  RTFREZ -- Lock job in memory without repositioning
                                ;-------------------------------------------------------------------
                                ;  The RTFREZ emt locked a job in memory without repositioning it.
                                ;
000460  032767  000000G 000000G RTFREZ: BIT     #PO$LOK,PRIVCO  ;Do we have privilege to do this?
000466  001001                        BNE     20$             ;Br if yes
000470  005000                        CLR     R0              ;Return error code 0
000472  052761  000000G 000000G 20$:    BIS     #$MLOCK,LSW6(R1);LOCK JOB IN MEMORY
000500  000167  000000G                 JMP     EMTXIT          ;FINISHED

                                        .SBTTL  RTUNLK -- Unlock job from memory
                                ;-------------------------------------------------------------------
                                ;  The RTUNLK emt unlockes a job from memory.
                                ;
000504  042761  000000G 000000G RTUNLK: BIC     #$MLOCK,LSW6(R1);UNLOCK FROM MEMORY
000512  000167  000000G                 JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  RTPEEK -- Peek at I/O page
                                ;-------------------------------------------------------------------------
                                ;  RTPEEK emt is used to access a single word in the I/O page.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of cell in I/O page to be accessed.
                                ;
                                ;  Outputs:
                                ;   Contents of cell are returned to user in R0.
                                ;
000516  004767  001250          RTPEEK: CALL    PRVMEM          ;Can we access physical memory?
000522  017767  000002G 000000G       MOV     @EMTBLK+2,UR0   ;PEEK AT I/O PAGE
000530  000421                        BR      RTXIT

                                        .SBTTL  RTPOKE -- Poke into I/O page
                                ;-------------------------------------------------------------------------
                                ;  RTPOKE emt is used to store a word into a cell in the I/O page.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of cell in I/O page to be accessed.
                                ;   Arg2 = Data value to be stored into I/O page cell.
                                ;
000532  004767  001234          RTPOKE: CALL    PRVMEM          ;Can we access physical memory?
000536  016777  000004G 000002G       MOV     EMTBLK+4,@EMTBLK+2;STORE INTO CELL IN I/O PAGE
000544  000413                        BR      RTXIT

                                        .SBTTL  RTBIS  -- Bit set into I/O page
                                ;-------------------------------------------------------------------------
                                ;  RTBIS emt is used to do a bit set into a word in the I/O page.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of cell in I/O page to be accessed.
                                ;   Arg2 = Data value to be ORed into I/O cell.
                                ;
000546  004767  001220          RTBIS:  CALL    PRVMEM          ;Can we access physical memory?
000552  056777  000004G 000002G       BIS     EMTBLK+4,@EMTBLK+2;DO BIS INTO I/O PAGE CELL
000560  000405                        BR      RTXIT

                                        .SBTTL  RTBIC  -- Bit clear into I/O page
                                ;-------------------------------------------------------------------------
                                ;  RTBIC emt is used to do a bit clear into a word in the I/O page.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of cell in I/O page to be accessed.
                                ;   Arg2 = Data value to be used as a bit clear mask.
                                ;
000562  004767  001204          RTBIC:  CALL    PRVMEM          ;Can we access physical memory?
000566  046777  000004G 000002G       BIC     EMTBLK+4,@EMTBLK+2;DO BIC INTO I/O PAGE CELL
000574  000167  000000G         RTXIT: JMP     EMTXIT          ;FINISHED WITH EMT
```

```
                                        .SBTTL  RTIOMP -- Map par7 to I/O page
                                ;-------------------------------------------------------------------------
                                ;  RTIOMP is called to map the upper 8Kb (par7) of the job's virtual
                                ;  address space to the I/O page.
                                ;
000600  004767  001166          RTIOMP: CALL    PRVMEM          ;Can we access I/O page?
000604  052761  000000G 000000G        BIS     #$IOMAP,LSW6(R1);SET FLAG SAYING WE ARE ACCESSING I/O PAGE
000612  004767  000000G         RTMAP:  CALL    SETMAP          ;RELOAD MAPPING REGISTERS FOR JOB
000616  000766                          BR      RTXIT

                                        .SBTTL  RTRMMP -- Map par7 to simulated Rmon
                                ;-------------------------------------------------------------------------
                                ;  RTRMMP emt is used to map the par7 region of the user's job back to
                                ;  the simulated Rmon.
                                ;
000620  042761  000000G 000000G RTRMMP: BIC     #$IOMAP,LSW6(R1);SAY WE ARE NO LONGER MAPPED TO I/O PAGE
000626  000771                          BR      RTMAP           ;RELOAD MAPPING REGISTERS FOR JOB
```

```
                                        .SBTTL  .SPND  -- Suspend job's execution
                                ;-------------------------------------------------------------------
                                ;  The .SPND emt is used to suspend the job's execution until a .RSUM
                                ;  is done.
                                ;
000630  005361  000000G       RTSPND: DEC     LSPND(R1)       ;DEC SUSPENSION COUNTER FOR JOB
000634  002023                        BGE     1$              ;BR IF WE SHOULD NOT SUSPEND JOB
                                ;  Don't suspend job if running in a completion routine.
000636  105767  000000G              TSTB    CURCP           ;ARE WE RUNNING IN A COMPLETION ROUTINE NOW?
000642  001020                        BNE     1$              ;BR IF YES
                                ;
                                ;  Suspend execution of the job until a .RSUM emt is done.
                                ;
000644                          3$:     DISABL                  ;;;DISABLE INTERRUPTS
000652  005761  000000G              TST     LSPND(R1)       ;;;HAS RESUME BEEN ISSUED?
000656  002007                        BGE     4$              ;;;BR IF WE SHOULD RESTART JOB
000660  012700  000000G              MOV     #S$SPND,R0      ;;;JOB SUSPENDED STATE
000664  004767  000000G              CALL    QHDSPN          ;;;SUSPEND THE JOB ** ENABLES INTERRUPTS **
000670  004767  000000G              CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
000674  000763                        BR      3$              ;ATTEMPT TO RESUME THE JOB
                                ;
                                ;  Finished doing the .spnd -- resume the job's execution.
                                ;
000676                          4$:     ENABL                   ;;;ENABLE INTERRUPTS
000704  000167  000000G        1$:     JMP     EMTXIT          ;FINISHED WITH .SPND

                                        .SBTTL  .RSUM  -- Resume job's execution
                                ;-------------------------------------------------------------------
                                ;  The .RSUM emt is used to resume execution of a job that is asleep
                                ;  because of doing a .SPND.
                                ;
000710  005261  000000G       RTRSUM: INC     LSPND(R1)       ;INC JOB'S .SPND COUNTER
000714  000167  000000G              JMP     EMTXIT          ;THAT'S ALL WE HAVE TO DO
```

```
                                        .SBTTL  RTCMPL -- Schedule a completion routine
                                ;-----------------------------------------------------------------------
                                ;  The RTCMPL EMT is used to schedule a completion routine for the job.
                                ;
                                ;  The form of the EMT is
                                ;
                                ;       EMT     375
                                ;
                                ;  With R0 pointing to the following EMT argument block:
                                ;
                                ;       .BYTE   21,140
                                ;       .WORD   completion_routine_address
                                ;       .WORD   completion_routine_priority
                                ;       .WORD   value_to_pass_in_R1
                                ;       .WORD   0
                                ;
000720                          RTCMPL:
                                ;
                                ;  Get a completion routine queue element
                                ;
000720  004767  000000G                 CALL    GETQ            ;Get a free completion queue element
                                ;
                                ;  Set up the queue element
                                ;
000724  010104                          MOV     R1,R4           ;Carry queue element pointer in R4
000726  116701  000000G                 MOVB    CORUSR,R1       ;Get job index number
000732  110164  000000G                 MOVB    R1,CQ$JOB(R4)   ;Set job number in queue element
000736  012703  000002G                 MOV     #EMTBLK+2,R3    ;Get pointer to block with EMT arguments
000742  012364  000000G                 MOV     (R3)+,CQ$RTN(R4);Set address of completion routine
000746  012302                          MOV     (R3)+,R2        ;Get execution priority value
000750  001012                          BNE     2$              ;Br if priority not = 0
000752  112764  000000G 000000G         MOVB    #S$TWFN,CQ$RNS(R4);Set non-real-time execution state
000760  116164  000000G 000000G         MOVB    LBSPRI(R1),CQ$PRI(R4);Set execution priority
000766  112764  000000G 000000G         MOVB    #CP$STD,CQ$CP(R4);Set standard compl rtn class priority
000774  000420                          BR      4$
000776  112764  000000G 000000G 2$:     MOVB    #CP$RT,CQ$CP(R4);Set real-time completion class priority
001004  112764  000000G 000000G         MOVB    #S$RT,CQ$RNS(R4);Set real-time execution state
001012  116700  000000G                 MOVB    VPRIHI,R0       ;Get base real-time priority
001016  060002                          ADD     R0,R2           ;Add base real-time priority
001020  020227  000000G                 CMP     R2,#MAXPRI      ;Make sure we don't overflow
001024  101402                          BLOS    3$              ;Br if ok
001026  012702  000000G                 MOV     #MAXPRI,R2      ;Truncate to max allowed
001032  110264  000000G         3$:     MOVB    R2,CQ$PRI(R4)   ;Set execution priority value
001036  012364  000000G         4$:     MOV     (R3)+,CQ$R1(R4) ;Set value to pass to completion routine in R1
001042  005064  000000G                 CLR     CQ$R0(R4)       ;Pass 0 in R0
001046  013764  000000G 000000G         MOV     @#KPAR5,CQ$PA5(R4);Save current kernel PAR 5 mapping
                                ;
                                ;  Queue the completion routine request
                                ;
001054  004767  000000G                 CALL    QCOMPL          ;Queue the completion routine
                                ;
                                ;  Finished
                                ;
001060  000167  000000G                 JMP     EMTXIT          ;Finished with EMT
```

```
                                        .SBTTL  RTEXJB -- Gain exclusive use of system
                                ;-------------------------------------------------------------------
                                ;  The RTEXJB EMT is used to obtain exclusive use of the system
                                ;  by a real-time job.  All other jobs are suspended until the RTALJB
                                ;  EMT is used to release exclusive access.
                                ;
001064  004767  000662          RTEXJB: CALL    PRVRT           ;Do we have Real-time privilege?
                                ;
                                ;  Set flag that says this job has exclusive access to the system
                                ;
001070  110167  000000G             MOVB    R1,EXCJOB       ;SAY THIS JOB HAS EXCLUSIVE ACCESS
                                ;
                                ;  Finished
                                ;
001074  000167  000000G             JMP     EMTXIT          ;EXIT FROM EMT

                                        .SBTTL  RTALJB -- Release exclusive access claim to system
                                ;-------------------------------------------------------------------
                                ;  The RTALJB emt is used to release exclusive access to the system.
                                ;
001100  105067  000000G         RTALJB: CLRB    EXCJOB          ;SAY WE NO LONGER HAVE EXCLUSIVE ACCESS
                                ;
                                ;  Set flag to cause execution scheduler to be called
                                ;
001104  105267  000000G             INCB    DOSCHD          ;CALL THE SCHEDULER LATER
                                ;
                                ;  Finished
                                ;
001110  000167  000000G             JMP     EMTXIT          ;EXIT FROM EMT
```

```
                                        .SBTTL  RTSPL  -- Set processor priority level
                                ;-----------------------------------------------------------------------
                                ;  The RTSPL EMT is used to set the processor priority level that
                                ;  will be used while running in user mode.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Processor priority level (0 to 7).
                                ;
001114  004767  000632          RTSPL:  CALL    PRVRT           ;Do we have real-time privilege?
001120  016702  0000020G               MOV     EMTBLK+2,R2     ;GET REQUESTED PROCESSOR PRIORITY LEVEL
001124  042702  177770                 BIC     #177770,R2      ;CLEAR ALL BUT PRIORITY LEVEL FIELD
001130  072227  000005                 ASH     #5,R2           ;POSITION TO PRIORITY FIELD IN PS
001134  042767  000340  0000000G       BIC     #340,EMTPS      ;CLEAR OLD PRIORITY FIELD IN PS
001142  050267  0000000G               BIS     R2,EMTPS        ;STORE NEW PRIORITY LEVEL INTO PS
001146  000167  0000000G               JMP     EMTXIT          ;DO EMT EXIT WHICH WILL LOAD PS
```

```
                                        .SBTTL  RTCVEC -- Connect completion routine to interrupt vector
                                ;-----------------------------------------------------------------------
                                ;  RTCVEC emt is used to connect an interrupt vector to a user completion
                                ;  routine.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of interrupt vector.
                                ;   Arg2 = Address of completion routine.
                                ;   Arg3 = Priority of completion routine (0 to 7).
                                ;
                                ;  Entry point for directly connected interrupt service routine
                                ;
001152  004767  000574          RTDVEC: CALL    PRVRT           ;Do we have real-time privilege?
001156  032761  000000G 000000G        BIT     #$MLOCK,LSW6(R1);IS JOB LOCKED IN MEMORY?
001164  001004                        BNE     1$              ;BR IF YES
001166  012700  000003                MOV     #3,R0           ;RETURN EMT ERROR CODE 3 IF NOT
001172  000167  000000G               JMP     SETERR
001176  012705  000000G          1$:   MOV     #VF$DIR,R5      ;SET DIRECTLY-CONNECTED INTERRUPT FLAG
001202  000401                        BR      RTXVEC          ;ENTER COMMON CODE
                                ;
                                ;  Entry point for completion routine type service routine
                                ;
001204  005005                  RTCVEC: CLR     R5              ;CLEAR DIRECTLY-CONNECTED INTERRUPT FLAG
                                ;
                                ;  See if interrupt vector is already connected to a service routine
                                ;
001206  016700  000002G         RTXVEC: MOV     EMTBLK+2,R0     ;GET ADDRESS OF VECTOR WE ARE CONNECTING TO
001212  006200                          ASR     R0              ;DROP LOW-ORDER BIT
001214  010003                          MOV     R0,R3           ;SAVE VECTOR VALUE
001216  004767  000570                  CALL    SRCVEC          ;SEE IF THERE IS A VECTOR CONTROL BLOCK ALREADY
001222  103407                          BCS     1$              ;BR IF NO CONTROL BLOCK FOR THIS VECTOR
001224  126201  000000G                 CMPB    VC$JOB(R2),R1   ;IS OUR JOB THE ONE THAT OWNS THE VECTOR?
001230  001414                          BEQ     2$              ;BR IF YES
001232  012700  000002                  MOV     #2,R0           ;ERROR CODE 2 MEANS SOMEONE ELSE OWNS VECTOR
001236  000167  000000G                 JMP     SETERR
                                ;
                                ;  Vector is free.  Get a vector control block and set it up.
                                ;
001242  005000                  1$:     CLR     R0              ;GET A FREE VECTOR CONTROL BLOCK
001244  004767  000542                  CALL    SRCVEC
001250  103004                          BCC     2$              ;BR IF GOT ONE
001252  012700  000001                  MOV     #1,R0           ;NO FREE VECTOR CONTROL BLOCKS
001256  000167  000000G                 JMP     SETERR
                                ;
                                ;  Found a free vector control block (Address is in R2).
                                ;  Set it up.
                                ;
001262  110162  000000G         2$:     MOVB    R1,VC$JOB(R2)   ;SET JOB NUMBER
001266  110362  000000G                 MOVB    R3,VC$VEC(R2)   ;SET ADDRESS OF VECTOR
001272  110562  000000G                 MOVB    R5,VC$FLG(R2)   ;SET CONTROL FLAGS (VF$xxx)
001276  016762  000004G 000000G        MOV     EMTBLK+4,VC$RTN(R2);SET ADDRESS OF COMPLETION ROUTINE
001304  016700  000006G                 MOV     EMTBLK+6,R0     ;GET COMPLETION ROUTINE PRIORITY
001310  042700  177770                  BIC     #^C7,R0         ;FORCE TO RANGE 0-7
001314  110062  000000G                 MOVB    R0,VC$PRI(R2)   ;SAVE PRIORITY IN VECTOR CONTROL BLOCK
                                ;
                                ;  Make vector point to vector control block.
                                ;
```

```
     58 001320  006303                          ASL     R3              ;GET ADDRESS OF VECTOR
     59 001322  010223                          MOV     R2,(R3)+        ;SET VECTOR PC -- SEND INT TO INT CONTROL BLOCK
     60 001324  012713  000340                   MOV     #340,(R3)       ;SET VECTOR PS
     61                                 ;
     62                                 ;  Finished
     63                                 ;
     64 001330  000167  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  RTRVEC -- Release interrupt vector
                                ;-------------------------------------------------------------------------
                                ;  RTRVEC emt is used to release an interrupt vector connection.
                                ;
                                ;  Inputs:
                                ;   Arg1 = Address of vector.
                                ;
001334  004767  000412          RTRVEC: CALL    PRVRT           ;Do we have real-time privilege?
001340  016700  0000026G              MOV     EMTBLK+2,R0     ;GET ADDRESS OF VECTOR
001344  006200                       ASR     R0              ;DROP LOW-ORDER BIT
001346  004767  000440               CALL    SRCVEC          ;SEARCH FOR VECTOR CONTROL BLOCK
001352  103405                       BCS     1$              ;BR IF NONE FOR THIS VECTOR
001354  120162  0000006G             CMPB    R1,VC$JOB(R2)   ;DO WE OWN THE VECTOR?
001360  001002                       BNE     1$              ;BR IF NOT
001362  004767  000004               CALL    RELVEC          ;RELEASE THE VECTOR
001366  000167  0000006G     1$:     JMP     EMTXIT

                                ;-------------------------------------------------------------------------
                                ;  RELVEC is called to release a user's link with an interrupt vector.
                                ;  The vector is set to point to the unexpected interrupt routine and
                                ;  the vector control block is freed.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of vector control block.
                                ;
001372  010346                  RELVEC: MOV     R3,-(SP)
                                ;
                                ;  If .DEVICE list already specified a reset address for this vector,
                                ;  don't make it point to normal unexpected-interrupt location.
                                ;
001374  132762  0000006G 0000006G       BITB    #VF$DET,VC$FLG(R2);Did .DEVICE disconnect the interrupt?
001402  001031                       BNE     2$              ;Br if yes
                                ;
                                ;  Set interrupt vector to point to unexpected interrupt routine.
                                ;
001404  005003                       CLR     R3              ;GET ADDRESS OF INT VECTOR FROM VECTOR CONTROL
001406  156203  0000006G             BISB    VC$VEC(R2),R3
001412  006303                       ASL     R3              ;CONVERT TO REAL ADDRESS
001414  010300                       MOV     R3,R0
001416  105767  0000006G             TSTB    VUXIFL          ;SHOULD WE CRASH OR IGNORE UNEXPECTED INTS?
001422  001005                       BNE     1$              ;BR IF WE SHOULD CRASH
001424  012720  0000006G             MOV     #UEXRTN,(R0)+   ;SET PC TO ROUTINE TO IGNORE INTS
001430  012710  000340               MOV     #340,(R0)       ;SET PS PRIO=7 FOR INTERRUPT
001434  000414                       BR      2$
001436  012720  0000006G     1$:     MOV     #UEXINT,(R0)+   ;SET VECTOR TO GO TO UNEXPECTED INT ROUTINE
                                ;
                                ;  Set PS in interrupt vector to encode interrupt address.
                                ;
001442  072327  177776               ASH     #-2,R3          ;DROP LOW-ORDER 2 BITS OF VECTOR ADDRESS
001446  010346                       MOV     R3,-(SP)
001450  042716  177760               BIC     #^C17,(SP)      ;CLEAR ALL BUT N-Z-V-C FIELDS
001454  006303                       ASL     R3              ;SHIFT OVER ADDRESS TO AVOID T FLAG
001456  042703  177437               BIC     #^C340,R3       ;CLEAR ALL BUT PRIORITY FIELD
001462  052603                       BIS     (SP)+,R3        ;COMBINE N-Z-V-C WITH PRIO FIELD
001464  010310                       MOV     R3,(R0)         ;STORE INTO VECTOR PS WORD
                                ;
                                ;  Free the vector control block
```

```
      58                                        ;
      59 001466  105062  0000006       2$:     CLRB    VC$JOB(R2)
      60 001472  105062  0000006               CLRB    VC$VEC(R2)
      61                                        ;
      62                                        ;  Finished
      63                                        ;
      64 001476  012603                         MOV     (SP)+,R3
      65 001500  000207                         RETURN
```

```
                                        .SBTTL  RTDEV  -- .DEVICE Emt
                                ;-----------------------------------------------------------------------
                                ;  The .DEVICE emt is used to declare a list of addresses and values to
                                ;  be stored when a job exits.
                                ;
001502  004767  000244          RTDEV:  CALL    PRVRT           ;Do we have real-time privilege?
                                ;
                                ;  Determine if this is a simple or linked type list
                                ;
001506  105767  000000G        1$:     TSTB    EMTBLK          ;SIMPLE OR LINKED LIST?
001512  001004                          BNE     2$              ;BR IF LINKED LIST
                                ;
                                ;  Simple (non-linked) list
                                ;
001514  016767  000002G 000000G        MOV     EMTBLK+2,DEVLS  ;SET ADDRESS OF .DEVICE LIST
001522  000407                          BR      9$              ;FINISHED
                                ;
                                ;  Linked list
                                ;
001524  016746  000000G        2$:     MOV     DEVLL,-(SP)     ;PUSH ADDRESS OF CURRENT LIST HEAD
001530  106677  000002G                MTPD    @EMTBLK+2       ;STORE AS FORWARD LINK FROM NEW LIST
001534  016767  000002G 000000G        MOV     EMTBLK+2,DEVLL  ;SAVE ADDRESS OF NEW LIST
                                ;
                                ;  Finished
                                ;
001542  000167  000000G        9$:     JMP     EMTXIT
```

```
                                        .SBTTL  RTSTOP -- Real-time cleanup at end of job execution
                                ;-----------------------------------------------------------------------
                                ;  RTSTOP is called when a job exits for any reason.
                                ;  It does any necessary real-time cleanup for the job.
                                ;  This consists of the following things:
                                ;   1. Process any specified .DEVICE list.
                                ;   2. Disconnect any interrupt vectors attached to the job.
                                ;   3. Release exclusive access of system if job has gotten it.
                                ;
                                ;  Inputs:
                                ;   R1 = Current job number
                                ;
001546  010246                  RTSTOP: MOV     R2,-(SP)
                                ;
                                ;  If job did a .DEVICE emt, process the device reset list.
                                ;
                                ;  Process linked type lists first
001550  016702  000000G         1$:     MOV     DEVLL,R2        ;IS THERE A LINKED TYPE LIST TO PROCESS?
001554  001415                          BEQ     3$              ;BR IF NOT
001556  005067  000000G                 CLR     DEVLL           ;CLEAR IN CASE WE TRAP
001562  106522                          MFPD    (R2)+           ;GET ADDRESS OF FORWARD LINK FROM LIST HEAD
001564  012667  000000G                 MOV     (SP)+,DEVLL     ;SAVE THIS ADDRESS AS NEW HEAD OF LINKED LIST
001570  106522                  2$:     MFPD    (R2)+           ;GET ADDRESS TO STORE INTO
001572  012600                          MOV     (SP)+,R0
001574  001765                          BEQ     1$              ;BR IF REACHED END OF LIST
001576  106522                          MFPD    (R2)+           ;GET VALUE TO STORE INTO CELL
001600  012610                          MOV     (SP)+,(R0)      ;DO THE STORE
001602  004767  000112                 CALL    CKVREL          ;See if we just released an interrupt conctn.
001606  000770                          BR      2$              ;GO PROCESS NEXT ITEM IN LIST
                                ;  Now process non-linked type lists
001610  016702  000000G         3$:     MOV     DEVLS,R2        ;ANY NON-LINKED .DEVICE LIST?
001614  001412                          BEQ     INTSTP          ;BR IF NOT
001616  005067  000000G                 CLR     DEVLS           ;CLEAR IN CASE WE TRAP
001622  106522                  4$:     MFPD    (R2)+           ;GET ADDRESS TO STORE INTO
001624  012600                          MOV     (SP)+,R0
001626  001405                          BEQ     INTSTP          ;BR IF REACHED END OF LIST
001630  106522                          MFPD    (R2)+           ;GET VALUE TO STORE
001632  012610                          MOV     (SP)+,(R0)      ;DO THE STORE
001634  004767  000060                 CALL    CKVREL          ;See if we released an interrupt connection
001640  000770                          BR      4$              ;GO PROCESS NEXT ITEM IN LIST
                                ;
                                ;  Disconnect all interrupt vectors associated with this job
                                ;
001642  012702  000000G         INTSTP: MOV     #VCBBAS,R2      ;POINT TO 1ST VECTOR CONTROL BLOCK
001646  020227  000000G         1$:     CMP     R2,#VCBEND      ;CHECKED ALL VECTOR CONTROL BLOCKS?
001652  103010                          BHIS    4$              ;BR IF YES
001654  120162  000000G                 CMPB    R1,VC$JOB(R2)   ;IS THIS VCB IN USE BY THIS JOB?
001660  001002                          BNE     2$              ;BR IF NOT
001662  004767  177504                 CALL    RELVEC          ;RELEASE THE VECTOR AND THE VCB
001666  062702  000000G         2$:     ADD     #VC$$SZ,R2      ;POINT TO NEXT VCB
001672  000765                          BR      1$
                                ;
                                ;  If job has exclusive access to the system, release it
                                ;
001674  120167  000000G         4$:     CMPB    R1,EXCJOB       ;DOES THIS JOB HAVE EXCLUSIVE ACCESS TO SYS?
001700  001005                          BNE     3$              ;BR IF NOT
001702  105767  000000G                 TSTB    CINFLG          ;IS THIS JOB DOING A CHAIN NOW?
```

```
     58 001706  001002                          BNE     3$              ;IF YES THEN RETAIN EXCLUSIVE ACCESS
     59 001710  105067  000000G                  CLRB    EXCJOB          ;RELEASE EXCLUSIVE ACCESS
     60                                ;
     61                                ;  Finished
     62                                ;
     63 001714  012602                 3$:     MOV     (SP)+,R2
     64 001716  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  See if the cell being stored into by the .DEVICE list is an interrupt
                                ;  vector.  If so, mark the vector control block so that we don't alter
                                ;  the interrupt location later.
                                ;
                                ;  Inputs:
                                ;   R0 = Address of cell being modified by .DEVICE list processing.
                                ;
001720  010246                 CKVREL: MOV     R2,-(SP)
                                ;
                                ;  See if location being modified is an interrupt vector
                                ;
001722  020027  000500                  CMP     R0,#500         ;Could this be an interrupt vector?
001726  103007                          BHIS    9$              ;Br if not
001730  006200                          ASR     R0              ;Get vector address / 2
001732  004767  000054                  CALL    SRCVEC          ;Try to find vector control block
001736  103403                          BCS     9$              ;Br if no vector attachment
                                ;
                                ;  Set flag in vector control block which prevents us from connecting
                                ;  this vector to unexpected-interrupt location later.
                                ;
001740  152762  000000G 000000G        BISB    #VF$DET,VC$FLG(R2) ;Say vector has been disconnected
                                ;
                                ;  Finished
                                ;
001746  012602                  9$:     MOV     (SP)+,R2
001750  000207                          RETURN
```

```
                                        .SBTTL  ** Subroutines **
                                        .SBTTL  PRVRT  -- Determine if job has real-time privilege
                                ;-------------------------------------------------------------------------
                                ;  Determine if the job has real-time privilege.
                                ;  If not, error code 0 is returned for the EMT.
                                ;
001752  032767  000000G 000000G PRVRT:  BIT     #PO$RT,PRIVCO   ;Does job have real-time privilege?
001760  001003                         BNE     9$              ;Br if yes
001762  005000                         CLR     R0              ;Return error code 0
001764  000167  000000G                JMP     SETERR
001770  000207                 9$:     RETURN

                                        .SBTTL  PRVMEM -- Check for access authorization to phys memory
                                ;-------------------------------------------------------------------------
                                ;  Determine if the job can access physical memory.
                                ;  If not, error code 0 is returned for the EMT.
                                ;
001772  032767  000000G 000000G PRVMEM: BIT     #PO$MEM,PRIVCO  ;Can we access physical memory?
002000  001003                         BNE     9$              ;Br if yes
002002  005000                         CLR     R0              ;Return error code 0
002004  000167  000000G                JMP     SETERR
002010  000207                 9$:     RETURN
```

```
      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 002012  012702  0000000G          SRCVEC: MOV     #VCBBAS,R2      ;GET ADDRESS OF BASE OF CONTROL BLOCK AREA
     14 002016  020227  0000000G          1$:     CMP     R2,#VCBEND      ;CHECKED ALL BLOCKS?
     15 002022  103006                            BHIS    4$              ;BR IF YES
     16 002024  120062  0000000G                  CMPB    R0,VC$VEC(R2)   ;IS THIS CONTROL BLOCK FOR THE VECTOR?
     17 002030  001405                            BEQ     2$              ;BR IF YES
     18 002032  062702  0000000G                  ADD     #VC$$SZ,R2      ;POINT TO NEXT VECTOR CONTROL BLOCK
     19 002036  000767                            BR      1$
     20 002040  000261                    4$:     SEC                     ;SIGNAL FAILURE ON RETURN
     21 002042  000401                            BR      3$
     22 002044  000241                    2$:     CLC                     ;SIGNAL SUCCESS ON RETURN
     23 002046  000207                    3$:     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: 18176 Words  ( 71 Pages)
Operating  system: RT-11

Elapsed time:  00:00:20.11
,LP:TSRTX=DK:TSRTX/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$IOMAP    1-35      7-7       7-16
$MLOCK    1-38      5-8       5-15      12-14
$UDSPC    1-47      3-17
BADEMT    1-34      2-25
CHKABT    1-41      8-19
CINFLG    1-43     15-57
CKVREL   15-28     15-39     16-9#
CORUSR    1-48      9-26
CP$RT     1-46      9-36
CP$STD    1-46      9-34
CQ$CP     1-46      9-34*     9-36*
CQ$JOB    1-36      9-27*
CQ$PA5    1-42      9-46*
CQ$PRI    1-42      9-33*     9-43*
CQ$RO     1-36      9-45*
CQ$R1     1-47      9-44*
CQ$RNS    1-42      9-32*     9-37*
CQ$RTN    1-36      9-29*
CUPARO    1-45      1-47      4-50*
CUPDRO    1-45      1-47      4-65*
CURCP     1-41      8-9
DEVLL     1-42     14-20     14-22*    15-18     15-20*    15-22*
DEVLS     1-42     14-15*    15-31     15-33*
DOSCHD    1-43     10-25*
EMTBLK    1-34      2-21      3-45      3-55      4-14      4-30      4-36      4-57      4-60      6-12      6-24      6-24*
          6-36      6-36*     6-48      6-48*     9-28     11-10     12-27     12-51     12-52     13-9      14-10     14-15
         14-21*    14-22
EMTPS     1-37     11-13*    11-14*
EMTXIT    1-34      3-50      4-26      4-72      5-9       5-16      6-49      8-25      8-33      9-54      10-15     10-29
         11-15     12-64     13-16     14-26
EXCJOB    1-43     10-11*    10-21*    15-55     15-59*
GETQ      1-48      9-21
INTPRI    1-37      8-24
INTSTP   15-32     15-36     15-44#
KPAR5     1-46      9-46
LBSPRI    1-38      9-33
LPARBS    1-34      3-36
LSPND     1-41      8-6*      8-15      8-32*
LSW11     1-47      3-17
LSW2      1-34
LSW6      1-35      5-8*      5-15*     7-7*      7-16*     12-14
MAXPRI    1-48      9-40      9-42
MXRTFN    2-23      2-51#
OVRHC     1-39
PO$LOK    1-38      5-5
PO$MEM    1-40     17-18
PO$RT     1-40     17-7
PRIVCO    1-40      5-5       17-7      17-18
PRVMEM    4-13      6-11      6-23      6-35      6-47      7-6       17-18#
PRVRT    10-7      11-9      12-13     13-8      14-6      17-7#
PS        1-53#     8-14*     8-24*
QCOMPL    1-36      9-50
QHDSPN    1-42      8-18
RDAR      1-44      3-22
RDDR      1-44      3-20
RELVEC   13-15     13-26#    15-49
```

Cross reference table (CREF V05.05)

```
RPAR      1-44      3-30      4-20*     4-48*
RPDR      1-44      3-28      4-21*     4-63*
RTALJB    2-44     10-21#
RTBIC     2-35      6-47#
RTBIS     2-34      6-35#
RTCMPL    2-48      9-17#
RTCVAD    2-31      3-18      3-27#
RTCVC1    3-19      3-27      3-55#
RTCVC2    3-23      3-31#
RTCVC3    3-21      3-29      3-36#
RTCVC4    3-32      3-40#
RTCVDA    2-49      3-17#
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#     2-51
RTEXJB    2-43     10-7#
RTFREZ    2-42      5-5#
RTIOMP    2-36      7-6#
RTLOCK    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
```

Cross reference table (CREF V05.05)

Cross reference table (CREF V05.05)