Table of contents

```
                                        .TITLE  TSRT -- Resident portion of real-time code
                                        .ENABL  LC
                                        .DSABL  GBL
                                ;-----------------------------------------------------------------------
                                ;  This module contains the resident portion of the TSX-Plus real-time
                                ;  processing code.
                                ;
                                ;  Copyright (c) 1980, 1981, 1982, 1983.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee
                                ;  All rights reserved.
                                ;
000000                                  .CSECT  TSRT
000000                          TSRT:
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  RTINT
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  INTEN,FORK,VF$DIR,VC$FLG,GETRTQ,VC$JOB
                                        .GLOBL  CQ$JOB,VC$VEC,CQ$RO,VC$RTN,CQ$RTN
                                        .GLOBL  CQ$PA5,VC$PRI,CQ$PRI,QCOMPL,CQ$RNS
                                        .GLOBL  CURVC,LCXPAR,KPAR6,UPARO,CUPARO,UPDRO,CUPDRO
                                        .GLOBL  FPUUSE,UPMODE,PSW,UMSPSV,CPLEMT,UMODE,UPMODE
                                        .GLOBL  LSPND,FORCEX,CFLAG,EMTENT,CQ$R1,MAXPRI,VPRIHI,S$RT
                                        .GLOBL  LBSPRI,S$TWFN,FP$RT,KPAR5,MAPPAR
                                        .GLOBL  CQ$CP,CP$RT,CP$STD
                                ;
                                ;  Data areas
                                ;
000000  000000                  RISPSV: .WORD   0       ;Holds SP during direct interrupt service call
000002  000000                  VCHOLD: .WORD   0       ;Used to hold address of vec ctrl blk during .inten
```

```
                                        .SBTTL  RTINT  -- Real-time interrupt entry point
                                ;-------------------------------------------------------------------
                                ;  RTINT is the entry point for all real-time interrupts.
                                ;  The actual interrupt vector sent the interrupt to the vector control
                                ;  block for the interrupt which had the following instruction stored
                                ;  as its first word: [ JSR  R2,RTINT ]. Thus on entry to RTINT, R2
                                ;  points to byte 4 of the vector control block.
                                ;
                                ;  Save address of the vector control block.
                                ;
000004  162702  000004          RTINT:  SUB     #4,R2           ;GET ADDRESS OF START OF CONTROL BLOCK
000010  010267  177766                  MOV     R2,VCHOLD       ;SAVE ADDRESS ACROSS .INTEN
000014  012602                          MOV     (SP)+,R2        ;RESTORE R2
                                ;
                                ;  Now do a .INTEN and .FORK to get out of interrupt mode.
                                ;
000016  004567  0000000                 JSR     R5,INTEN        ;STANDARD INTERRUPT ENTRY
000022  000000                          .WORD   0               ;(PRIORITY 7)
000024  016704  177752                  MOV     VCHOLD,R4       ;PICK UP ADDRESS OF INTERRUPT CONTROL BLOCK
000030  004567  0000000                 JSR     R5,FORK         ;DROP DOWN TO FORK LEVEL
000034  0000000                         .WORD   FP$RT           ;Fork processing priority
                                ;
                                ;  At this point we are running at fork level with the address of the
                                ;  vector control block in R4.
                                ;
                                ;  Determine if interrupt is directly connected with interrupt
                                ;  service routine or it interrupt service routine is to be
                                ;  called as a completion routine.
                                ;
000036  132764  0000000 0000000         BITB    #VF$DIR,VC$FLG(R4);Is this a directly connected interrupt?
000044  001063                          BNE     DIENTR          ;Br if yes
                                ;
                                ;  This interrupt is connected to job through a completion routine.
                                ;  Set up a completion routine queue entry for the job connected with
                                ;  the interrupt.
                                ;
000046  004767  0000000                 CALL    GETRTQ          ;GET A COMPLETION QUEUE ENTRY
000052  116461  0000000 0000000         MOVB    VC$JOB(R4),CQ$JOB(R1) ;SET JOB NUMBER
000060  005000                          CLR     R0
000062  156400  0000000                 BISB    VC$VEC(R4),R0   ;GET VECTOR ADDRESS WITHOUT SIGN EXTENSION
000066  006300                          ASL     R0              ;CONVERT TO ABS ADDRESS
000070  010061  0000000                 MOV     R0,CQ$R0(R1)    ;PUT IN R0 WHEN WE ENTER COMPLETION ROUTINE
000074  016461  0000000 0000000         MOV     VC$RTN(R4),CQ$RTN(R1) ;SET ADDRESS OF COMPLETION ROUTINE
000102  013761  0000000 0000000         MOV     @#KPAR5,CQ$PA5(R1);Save current kernel PAR 5 mapping
000110  116405  0000000                 MOVB    VC$PRI(R4),R5   ;GET COMPLETION ROUTINE PRIORITY
000114  001014                          BNE     3$              ;Br if priority not zero
000116  112761  0000000 0000000         MOVB    #S$TWFN,CQ$RNS(R1);Set non-real-time execution state
000124  112761  0000000 0000000         MOVB    #CP$STD,CQ$CP(R1);Set standard compl rtn class priority
000132  116100  0000000                 MOVB    CQ$JOB(R1),R0   ;Get job index number
000136  116061  0000000 0000000         MOVB    LBSPRI(R0),CQ$PRI(R1);Set execution priority
000144  000417                          BR      4$
000146  112761  0000000 0000000 3$:     MOVB    #S$RT,CQ$RNS(R1);Set real-time execution state
000154  112761  0000000 0000000         MOVB    #CP$RT,CQ$CP(R1);Set real-time compl rtn class priority
000162  066705  0000000                 ADD     VPRIHI,R5       ;Add base real-time priority
000166  020527  0000000                 CMP     R5,#MAXPRI      ;Did priority overflow?
000172  101402                          BLOS    2$              ;Br if not
000174  012705  0000000                 MOV     #MAXPRI,R5      ;Reduce to maximum allowed
```

```
     58 000200  110561  0000000          2$:     MOVB    R5,CQ$PRI(R1)   ;Set execution priority
     59                                  ;
     60                                  ;  Queue a completion routine for the job.
     61                                  ;
     62 000204  010104                   4$:     MOV     R1,R4           ;GET ADDRESS OF COMPLETION QUEUE ENTRY
     63 000206  004767  0000000                  CALL    QCOMPL          ;QUEUE A COMPLETION REQUEST FOR THE JOB
     64                                  ;
     65                                  ;  Finished
     66                                  ;
     67 000212  000207                   1$:     RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Enter an interrupt service routine that is directly connected to
                                ;  a real-time interrupt.
                                ;
                                ;  Save pointer to interrupt control block
                                ;  (CURVC non-zero is also a flag that we are executing a directly
                                ;  connected interrupt service routine.)
                                ;
000214  010467  000000G       DIENTR: MOV     R4,CURVC        ;Save address of interrupt control block
                                ;
                                ;  Get index number of job that is to receive the interrupt and map
                                ;  KPAR6 to job's context block (note, the FORK saved original KPAR6).
                                ;
000220  116401  000000G               MOVB    VC$JOB(R4),R1   ;Get job index number
000224  016137  000000G 000000G       MOV     LCXPAR(R1),@#KPAR6 ;Map kernel par 6 to job context block
                                ;
                                ;  Save current user-mode PAR registers and load for new job
                                ;
000232  012701  000000G               MOV     #UPAR0,R1       ;Point to user PAR registers
000236  012702  000000G               MOV     #CUPAR0,R2      ;Point to PAR holding cells in context block
000242  012703  000000G               MOV     #UPDR0,R3       ;Point to user PDR registers
000246  012704  000000G               MOV     #CUPDR0,R4      ;Point to PDR holding cells in context block
000252  012700  000010                MOV     #8.,R0          ;There are 8 PAR and PDR register pairs
000256  011146                1$:     MOV     (R1),-(SP)      ;Save current user PAR value
000260  012221                        MOV     (R2)+,(R1)+     ;Set PAR to map to job being called
000262  011346                        MOV     (R3),-(SP)      ;Save current user PDR value
000264  012423                        MOV     (R4)+,(R3)+     ;Set PDR to map to job being called
000266  077005                        SOB     R0,1$           ;Loop if more PAR's to load
000270  016704  000000G               MOV     CURVC,R4        ;Recover pointer to vector control block
                                ;
                                ;  If Floating Point Unit is in use, save its status
                                ;
000274  105767  000000G               TSTB    FPUUSE          ;Is FPU in use?
000300  001414                        BEQ     2$              ;Br if not
000302  170202                        STFPS   R2              ;Save FPU status
000304  010246                        MOV     R2,-(SP)
000306  170011                        SETD                    ;Set to 64-bit mode
000310  174046                        STD     R0,-(SP)        ;AC0
000312  174146                        STD     R1,-(SP)        ;AC1
000314  174246                        STD     R2,-(SP)        ;AC2
000316  174346                        STD     R3,-(SP)        ;AC3
000320  172404                        LDD     R4,R0           ;AC4-->AC0
000322  174046                        STD     R0,-(SP)        ;(AC4)
000324  172405                        LDD     R5,R0           ;AC5-->AC0
000326  174046                        STD     R0,-(SP)        ;(AC5)
000330  170102                        LDFPS   R2              ;Leave FPU status same as on entry
                                ;
                                ;  Set up user-mode SP
                                ;
000332  052737  000000G 000000G 2$:     BIS     #UPMODE,@#PSW   ;Make sure previous-mode = user
000340  106506                        MFPD    SP              ;Save current user-mode stack pointer on stack
000342  016702  000000G               MOV     UMSPSV,R2       ;Should we use SP from context block?
000346  001001                        BNE     3$              ;Br if yes
000350  011602                        MOV     (SP),R2         ;No, use current user mode SP
                                ;
                                ;  Push address of completion exit EMT on user's stack
                                ;
```

```
     58 000352  012746  000000G          3$:     MOV     #CPLEMT,-(SP)   ;Address of exit emt in user's job
     59 000356  106642                           MTPD    -(R2)           ;Push onto user's stack
     60 000360  010246                           MOV     R2,-(SP)        ;Set user-mode SP for job being entered
     61 000362  106606                           MTPD    SP
     62                                 ;
     63                                 ;  Enter user's interrupt service routine
     64                                 ;
     65 000364  010667  177410                   MOV     SP,RISPSV       ;Save SP before entry to user's routine
     66 000370  012737  000410' 000030           MOV     #DIEMT,@#30     ;Send EMT traps to DIEMT routine
     67 000376  012746  000000C                  MOV     #UMODE!UPMODE,-(SP) ;Set PS for interrupt routine
     68 000402  016446  000000G                  MOV     VC$RTN(R4),-(SP);Set PC for interrupt routine
     69 000406  000002                           RTI                     ;Enter interrupt service routine
```

```
                                        .SBTTL  DIEMT  -- EMT executed from interrupt service routine
                                ;-------------------------------------------------------------------------
                                ;  An EMT was executed from within a directly connected real-time interrupt
                                ;  service routine.  This can only be one of three things:
                                ;    1. A request to return from the interrupt service routine.
                                ;    2. A request to schedule a completion routine for the job.
                                ;    3. A .RSUM EMT to restart the main-line program routine.
                                ;
000410                          DIEMT:
                                ;
                                ;  Determine if this is a request to exit from the interrupt service routine
                                ;
000410  021627  0000026               CMP     (SP),#CPLEMT+2  ;Is this the EMT to exit from int routine?
000414  001522                       BEQ     DIEXIT          ;Br if yes
                                ;
                                ;  This is not a request to exit from the interrupt service routine.
                                ;
000416  010046                       MOV     R0,-(SP)
000420  010146                       MOV     R1,-(SP)
000422  010246                       MOV     R2,-(SP)
000424  010446                       MOV     R4,-(SP)
                                ;
                                ;  See if it is a .RSUM EMT.
                                ;
000426  016704  0000006              MOV     CURVC,R4        ;Get address of vector control block
000432  016601  000010               MOV     8.(SP),R1       ;Get address past EMT instruction
000436  106541                       MFPD    -(R1)           ;Get the EMT instruction
000440  021627  104374               CMP     (SP),#104374    ;Could this be a .RSUM EMT?
000444  001012                       BNE     DICMPL          ;Br if not
000446  020027  001000               CMP     R0,#1000        ;Check contents of R0
000452  001103                       BNE     DIEXIT          ;Br if not .RSUM
                                ;
                                ;  Perform a .RSUM from within a directly connected interrupt routine
                                ;
000454  116401  0000006      DIRSUM: MOVB    VC$JOB(R4),R1   ;Get current job number
000460  005261  0000006              INC     LSPND(R1)       ;Increment .SPND count for job
000464  004767  0000006              CALL    FORCEX          ;Force execution of the job
000470  000463                       BR      DIEMTX          ;Go exit from EMT processing
```

```
 1                                      ;
 2                                      ;  Determine if this is an EMT to request that a completion routine
 3                                      ;  be scheduled for the job.
 4                                      ;
 5                                      ;  The form of the EMT is
 6                                      ;
 7                                      ;       EMT     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 000472  021627  104375          DICMPL: CMP     (SP),#104375    ;Is this an EMT 375 instruction?
18 000476  001071                          BNE     DIEXIT          ;Br if not
19 000500  106520                          MFPD    (R0)+           ;Get 1st word of EMT argument block
20 000502  022627  060021                  CMP     (SP)+,#60021    ;Function code = 140 and sub-fun = 21?
21 000506  001065                          BNE     DIEXIT          ;Br if not
22                                      ;
23                                      ;  This is a request to schedule a completion routine.
24                                      ;  Get a completion request queue element.
25                                      ;
26 000510  004767  0000000                 CALL    GETRTQ          ;Get a completion Q element (address in R1)
27                                      ;
28                                      ;  Set up the queue element
29                                      ;
30 000514  116461  0000000 0000000         MOVB    VC$JOB(R4),CQ$JOB(R1) ;Set job number for completion routine
31 000522  106520                          MFPD    (R0)+           ;Get address of completion routine
32 000524  012661  0000000                 MOV     (SP)+,CQ$RTN(R1);Set completion routine address in Q element
33 000530  106520                          MFPD    (R0)+           ;Get priority value
34 000532  012602                          MOV     (SP)+,R2
35 000534  001011                          BNE     2$              ;Br if priority value not zero
36 000536  112761  0000000 0000000       MOVB    #S$TWFN,CQ$RNS(R1);Set non-real-time execution state
37 000544  116102  0000000               MOVB    CQ$JOB(R1),R2   ;Get job number
38 000550  116261  0000000 0000000       MOVB    LBSPRI(R2),CQ$PRI(R1);Set execution priority
39 000556  000414                          BR      3$
40 000560  112761  0000000 0000000 2$:    MOVB    #S$RT,CQ$RNS(R1);Set real-time execution state
41 000566  066702  0000000               ADD     VPRIHI,R2       ;Add base priority for real-time jobs
42 000572  020227  0000000               CMP     R2,#MAXPRI      ;Did priority value overflow?
43 000576  101402                          BLOS    1$              ;Br if not
44 000600  012702  0000000               MOV     #MAXPRI,R2      ;Set maximum allowed priority
45 000604  110261  0000000         1$:   MOVB    R2,CQ$PRI(R1)   ;Set execution priority
46 000610  106520                  3$:   MFPD    (R0)+           ;Get value to be passed in R1
47 000612  012661  0000000               MOV     (SP)+,CQ$R1(R1) ;Set R1 value for completion routine
48 000616  005002                          CLR     R2
49 000620  156402  0000000               BISB    VC$VEC(R4),R2   ;Get vector address without sign extension
50 000624  006302                          ASL     R2              ;Convert to abs address
51 000626  010261  0000000               MOV     R2,CQ$R0(R1)    ;Pass to completion routine in R0
52                                      ;
53                                      ;  Queue the completion routine for the job
54                                      ;
55 000632  010104                          MOV     R1,R4           ;Get completion Q element pointer to R4
56 000634  004767  0000000                 CALL    @COMPL          ;Queue the completion routine request
57                                      ;
```

```
                                        ;  Return to interrupt service routine following EMT
                                        ;
000640  005726                  DIEMTX: TST     (SP)+           ;Remove EMT instruction from stack
000642  012604                          MOV     (SP)+,R4
000644  012602                          MOV     (SP)+,R2
000646  012601                          MOV     (SP)+,R1
000650  012600                          MOV     (SP)+,R0
000652  042766  0000000 000002         BIC     #CFLAG,2(SP)    ;C-flag will be reset on return from EMT
000660  000002                          RTI                     ;Return from .RSUM EMT
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Exit from directly connected interrupt service routine
                                ;
000662  016706  177112          DIEXIT: MOV     RISPSV,SP       ;Restore SP to state before entering int rtn
000666  106606                          MTPD    SP              ;Restore user-mode SP
                                ;
                                ;  Restore Floating Point Unit status if it is in use
                                ;
000670  105767  000000G                 TSTB    FPUUSE          ;Is FPU in use?
000674  001412                          BEQ     1$              ;Br if not
000676  170011                          SETD                    ;Set 64-bit mode
000700  172426                          LDD     (SP)+,R0        ;(AC5)
000702  174005                          STD     R0,R5           ;AC0--->AC5
000704  172426                          LDD     (SP)+,R0        ;(AC4)
000706  174004                          STD     R0,R4           ;AC0--->AC4
000710  172726                          LDD     (SP)+,R3        ;AC3
000712  172626                          LDD     (SP)+,R2        ;AC2
000714  172526                          LDD     (SP)+,R1        ;AC1
000716  172426                          LDD     (SP)+,R0        ;AC0
000720  170126                          LDFPS   (SP)+           ;FPU status register
                                ;
                                ;  Restore user-mode memory mapping
                                ;
000722  012701  0000200         1$:     MOV     #UPAR0+16.,R1   ;Point past last PAR register
000726  012702  0000200                 MOV     #UPDR0+16.,R2   ;Point past last PDR register
000732  012700  000010                  MOV     #8.,R0          ;Load 8 PAR/PDR register pairs
000736  012642                  2$:     MOV     (SP)+,-(R2)     ;Restore PDR register
000740  012641                          MOV     (SP)+,-(R1)     ;Restore PAR register
000742  077003                          SOB     R0,2$           ;Loop if more to restore
                                ;
                                ;  Say we are exiting from interrupt service routine
                                ;
000744  005067  0000000                 CLR     CURVC           ;Say not in interrupt service routine
000750  012737  0000000 000030          MOV     #EMTENT,@#30    ;Restore EMT trap vector
                                ;
                                ;  Return to FORK which will return from interrupt
                                ;
000756  000207                          RETURN                  ;Return to FORK
        000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  64 Words  ( 1 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:05.23
DK:TSRT,LP:TSRT=DK:TSRT.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

CFLAG	1-27	5-65			
CP$RT	1-29	2-53			
CP$STD	1-29	2-48			
CPLEMT	1-26	3-58	4-13		
CQ$CP	1-29	2-48*	2-53*		
CQ$JOB	1-23	2-38*	2-49	5-30*	5-37
CQ$PA5	1-24	2-44*			
CQ$PRI	1-24	2-50*	2-58*	5-38*	5-45*
CQ$RO	1-23	2-42*	5-51*		
CQ$R1	1-27	5-47*			
CQ$RNS	1-24	2-47*	2-52*	5-36*	5-40*
CQ$RTN	1-23	2-43*	5-32*		
CUPARO	1-25	3-20			
CUPDRO	1-25	3-22			
CURVC	1-25	3-9*	3-29	4-25	6-33*
DICMPL	4-29	5-17#			
DIEMT	3-66	4-9#			
DIEMTX	4-38	5-60#			
DIENTR	2-31	3-9#			
DIEXIT	4-14	4-31	5-18	5-21	6-4#
DIRSUM	4-35#				
EMTENT	1-27	6-34			
FORCEX	1-27	4-37			
FORK	1-22	2-20			
FP$RT	1-28	2-21			
FPUUSE	1-26	3-33	6-9		
GETRTQ	1-22	2-37	5-26		
INTEN	1-22	2-17			
KPAR5	1-28	2-44			
KPAR6	1-25	3-15*			
LBSPRI	1-28	2-50	5-38		
LCXPAR	1-25	3-15			
LSPND	1-27	4-36*			
MAPPAR	1-28				
MAXPRI	1-27	2-55	2-57	5-42	5-44
PSW	1-26	3-50*			
QCOMPL	1-24	2-63	5-56		
RISPSV	1-33#	3-65*	6-4		
RTINT	1-18	2-11#			
S$RT	1-27	2-52	5-40		
S$TWFN	1-28	2-47	5-36		
TSRT	1-14#				
UMODE	1-26	3-67			
UMSPSV	1-26	3-52			
UPARO	1-25	3-19	6-24		
UPDRO	1-25	3-21	6-25		
UPMODE	1-26	1-26	3-50	3-67	
VC$FLG	1-22	2-30			
VC$JOB	1-22	2-38	3-14	4-35	5-30
VC$PRI	1-24	2-45			
VC$RTN	1-23	2-43	3-68		
VC$VEC	1-23	2-40	5-49		
VCHOLD	1-34#	2-12*	2-19		
VF$DIR	1-22	2-30			
VPRIHI	1-27	2-54	5-41		