Table of contents

```
                                        .TITLE  TSTTY2 -- Auxiliary terminal control overlay
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
                                        .CSECT  TSTTY2
000000                          TSTTY2: .RAD50  /TT2/           ;Overlay region id
000000  100100                  ;
                                ;  TSTTY2 is an auxiliary terminal control overlay.
                                ;  The routines in TSTTY2 could logically go in TSTTY except
                                ;  that overlay has already reached the 8Kb size limit.
                                ;
                                ;  Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee USA
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  AUTSPD, DOSWIT, LINSWT, STTCPL, TTCPL, SPPRED
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  URO, QFREE, GETQ, S$TTSC, EMTMAP, CQ$PA5, LINPNT, DELCHR
                                        .GLOBL  LRTCHR, CVTLC, CQ$RO, S$IOFN, $NOINT, LSW7
                                        .GLOBL  $VBELL, EMTBLK, VALADW, LTTCR, EMTXIT
                                        .GLOBL  CQ$JOB, CQ$RNS, LPRI, CQ$PRI, CP$STD
                                        .GLOBL  CQ$CP, CQ$RTN, QCOMPL, LINNXT
                                        .GLOBL  LABTIM, $RBRK, LSW10, S600, SETSPD, LMXPRM, S9600, $NABRS
                                        .GLOBL  LSW9, $NOVLN, LSW2, $SUCF, $LOFCF, $NOIN, LSW3, LNPRIM
                                        .GLOBL  LSECPT, FSTSL, LSTSL, INITLN, LPARNT, LSECPT, $VNOTT, LSW
                                        .GLOBL  LBSPRI, VPRILO, VPRIHI, VPRIVR, LPRI, LWINDO, WINSF
                                        .GLOBL  LNMAP, $INIT, WINST, LOTSPC, LOTPNT, LOTBUF, LOTEND
                                        .GLOBL  TRNSTR, FORCEX, OVRHC, PR7, PSW, KPAR6, LTTPAR, INTPRI
                                        .GLOBL  S19200, S4800, S3600, S2400, S1800, S1200, S300, S150, S110
                                ;
                                ;  Macro definitions
                                ;
                                        .MACRO  DISABL          ;DISABLE INTERRUPTS
                                        BIS     #PR7, @#PSW
                                        .ENDM   DISABL

                                        .MACRO  ENABL           ;ENABLE INTERRUPTS
                                        BIC     INTPRI, @#PSW
                                        .ENDM   ENABL

                                        .MACRO  OCALL   ENTADD
                                        .IF     B, ENTADD
                                                .ERROR  ;OCALL without entry address
                                        .ENDC
                                        CALL    OVRHC
                                        .WORD   ENTADD
                                        .ENDM   OCALL
                                ;
                                ;  The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
                                ;  terminal character buffer area.  The previous contents of par6 map
                                ;  register are pushed on the stack and may be restored by using the
                                ;  UNMAP or UNMAPX macros.
                                ;  R1 must contain the line index number of the line whose buffers
```

```
                                ;  are being accessed.
                                ;  The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
                                ;  macros is that the X-versions are more efficient but may only be
                                ;  used from within interrupt service routines where we are guaranteed
                                ;  to be running on the system stack.
                                ;  The TTMAP and UNMAP versions of the macros must only be
                                ;  used in sections of code where the interrupts are disabled.
                                ;
                                        .MACRO  TTMAPX
                                        MOV     LTTPAR(R1),@#KPAR6
                                        .ENDM   TTMAPX


                                ;
                                ;
                                        .MACRO  UNMAPX
                                        .ENDM   UNMAPX
                                ;
                                        .MACRO  TTMAP
                                        MOV     @#KPAR6,MAPHLD
                                        MOV     LTTPAR(R1),@#KPAR6
                                        .ENDM   TTMAP
                                ;
                                        .MACRO  UNMAP
                                        MOV     MAPHLD,@#KPAR6
                                        .ENDM   UNMAP
                                ;
                                ;  Data areas
                                ;
000002  000000                  MAPHLD: .WORD   0               ;TEMP CELL USED BY TTMAP MACRO
```

```
                                        .SBTTL  AUTSPD -- Do autobaud speed selection for a line
                                ;-----------------------------------------------------------------
                                ;  AUTSPD is called when a character is received for a line that is not
                                ;  loged on and which is to have autobaud speed selection.
                                ;
                                ;  Inputs:
                                ;   R4 = Line index number.
                                ;   R5 = Character received.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Start this line now.
                                ;   C-flag set     ==> Do not start this line.
                                ;
000004  010146                  AUTSPD: MOV     R1,-(SP)
000006  010246                          MOV     R2,-(SP)
000010  010346                          MOV     R3,-(SP)
000012  010401                          MOV     R4,R1           ;Carry line index number in R1
000014  042705  177600                  BIC     #^C177,R5       ;Mask character to 7 bits
                                ;
                                ;  Ignore characters that come in too fast
                                ;
000020  026127  000000G 000142          CMP     LABTIM(R1),#98. ;Should we check this character?
000026  101055                          BHI     10$             ;Br if not
                                ;
                                ;  If we received a break (framing error), set the speed to 600 baud
                                ;  and get another character for autobaud detection.
                                ;
000030  032761  000000G 000000G         BIT     #$RBRK,LSW10(R1);Did we receive a break?
000036  001405                          BEQ     6$              ;Br if not
000040  012700  000000G                 MOV     #S600,R0        ;Select 600 baud
000044  004737  000000G                 CALL    SETSPD          ;Set line speed
000050  000436                          BR      8$              ;Don't start line yet
                                ;
                                ;  See if we should search speed table 1 or 2
                                ;
000052  012702  000230'         6$:     MOV     #ASTBL1,R2      ;Assume we are running at 9600 baud
000056  016103  000000G                 MOV     LMXPRM(R1),R3   ;Get current line speed
000062  000303                          SWAB    R3              ;Get to low-order byte
000064  042703  177760                  BIC     #^C17,R3        ;Clear all but speed code
000070  020327  000000G                 CMP     R3,#S9600       ;Is speed currently 9600?
000074  001411                          BEQ     1$              ;Br if yes
000076  012702  000256'                 MOV     #ASTBL2,R2      ;Assume speed is 600 baud
000102  020327  000000G                 CMP     R3,#S600        ;Is speed currently 600?
000106  001404                          BEQ     1$              ;Br if yes
                                ;
                                ;  Current speed is neither 9600 baud nor 600 baud.
                                ;  If character is carriage return, assume current speed is correct.
                                ;
000110  120527  000015                  CMPB    R5,#15          ;Is character carriage return?
000114  001427                          BEQ     11$             ;Br if yes -- speed must be correct
000116  000407                          BR      5$              ;Set speed to 9600 baud
                                ;
                                ;  See if we can find the received character in the autobaud table
                                ;
000120  105762  000001          1$:     TSTB    1(R2)           ;Have we reached the end of the table?
000124  001404                          BEQ     5$              ;Br if at end of table
000126  120522                          CMPB    R5,(R2)+        ;Does received char match this table entry?
```

```
                        BEQ     2$              ;Br if yes -- We found the speed
                        TSTB    (R2)+           ;Skip over speed code for this entry
                        BR      1$              ;Loop if more entries to check
                ;
                ;  We did not find a matching character in this table.
                ;  Reset speed to 9600 baud.
                ;
5$:             MOV     #S9600,R0       ;Select 9600 baud
                CALL    SETSPD          ;Set the line speed
                ;
                ;  We could not identify this character.
                ;  Start timer which will reset the baud rate to 9600 after 10 seconds.
                ;
8$:             MOV     #100.,LABTIM(R1);Start autobaud timer for this line
                BIS     #$NABRS,LSW9(R1);Reset speed to 9600 baud after time interval
10$:            SEC                     ;Say not to start the line this time
                BR      9$              ;Finished
                ;
                ;  We found the correct speed entry for this line
                ;
2$:             MOVB    (R2)+,R0        ;Get the correct speed code
                CALL    SETSPD          ;Set the line speed
                ;
                ;  Set a short timer to cause us to ignore any garbage characters
                ;  that may be received after the autobaud character.
                ;
11$:            MOV     #2.,LABTIM(R1)  ;Ignore input chars for 0.2 seconds
                BIC     #$NABRS,LSW9(R1);Clear need-reset flag
                CLC                     ;Signal to start the line
                ;
                ;  Finished
                ;
9$:             BIC     #$RBRK,LSW10(R1);Clear break-received flag for line
                MOV     (SP)+,R3
                MOV     (SP)+,R2
                MOV     (SP)+,R1
                RETURN
```

```
                                        ;
                                        ;  Table to indicate speed when we are receiving at 9600 baud
                                        ;
000230      177     0000        ASTBL1: .BYTE   177,S19200
000232      176     0000                .BYTE   176,S19200
000234      175     0000                .BYTE   175,S19200
000236      172     0000                .BYTE   172,S19200
000240      171     0000                .BYTE   171,S19200
000242      162     0000                .BYTE   162,S19200
000244      015     0000                .BYTE   015,S9600
000246      146     0000                .BYTE   146,S4800
                                        ;;      .BYTE   014,S3600
                                        ;;      .BYTE   176,S2400
000250      170     0000                .BYTE   170,S2400
                                        ;;      .BYTE   160,S1800
000252      000     0000                .BYTE   000,S1200
000254      000     000                 .BYTE   0,0
                                        ;
                                        ;  Table to indicate speed when we are receiving at 600 baud
                                        ;
000256      177     0000        ASTBL2: .BYTE   177,S2400
000260      176     0000                .BYTE   176,S1200
000262      172     0000                .BYTE   172,S1200
000264      171     0000                .BYTE   171,S1200
000266      170     0000                .BYTE   170,S1200
000270      162     0000                .BYTE   162,S1200
                                        ;;      .BYTE   015,S600
000272      146     0000                .BYTE   146,S300
                                        ;;      .BYTE   170,S150
000274      160     0000                .BYTE   160,S110
000276      140     0000                .BYTE   140,S110
000300      000     000                 .BYTE   0,0
```

```
                        .SBTTL  DOSWIT -- Switch to a virtual line
                ;-----------------------------------------------------------------
                ;  DOSWIT is called when we receive a control-W-digit sequence.
                ;
                ;  Inputs:
                ;   R1 = Current virtual line index number.
                ;   R5 = Binary value of virtual line selected.
                ;
000302  010246  DOSWIT: MOV     R2,-(SP)
000304  010346         MOV     R3,-(SP)
000306  010446         MOV     R4,-(SP)
                ;
                ;  See if this job is allowed to use virtual lines
                ;
000310  032761  0000000 0000000         BIT     #$NOVLN,LSW2(R1);Can job use virtual lines?
000316  001060         BNE     20$             ;Br if not
000320  032761  0000000 0000000         BIT     #<$SUCF!$LOFCF>,LSW9(R1);Doing logon/logoff processing?
000326  001054         BNE     20$             ;Br if yes
000330  032761  0000000 0000000         BIT     #$NOIN,LSW3(R1) ;Doing logon/logoff processing?
000336  001050         BNE     20$             ;Br if yes -- don't allow line switching
                ;
                ;  Determine if we are switching to a virtual line or back to the primary
                ;
000340  005705         TST     R5              ;Switching to primary line?
000342  001444         BEQ     10$             ;Br if switching to primary line
                ;
                ;  We are switching to a virtual line
                ;  See if line has been allocated yet.
                ;
000344  016102  0000000         MOV     LNPRIM(R1),R2   ;Get primary line #
000350  016203  0000000         MOV     LSECPT(R2),R3   ;Point to table of secondary line numbers
000354  060503         ADD     R5,R3           ;Point to entry for selected line
000356  105743         TSTB    -(R3)           ;Has this virtual line been allocated yet?
000360  001035         BNE     10$             ;Br if yes
                ;
                ;  Request to switch to a virtual line which has not been
                ;  allocated yet.
                ;  See if we can find a free virtual line.
                ;
000362  012703  0000000         MOV     #FSTSL,R3       ;Get # of first virtual line
000366  020327  0000000 3$:     CMP     R3,#LSTSL       ;Checked all?
000372  101032         BHI     20$             ;Br if yes
000374  005763  0000000         TST     LNPRIM(R3)      ;Is this line available?
000400  001403         BEQ     4$              ;Br if yes
000402  062703  000002          ADD     #2,R3           ;Get index to next virtual line
000406  000767         BR      3$
                ;
                ;  We have found a free virtual line.
                ;  It's index is in R3.
                ;  Initialize it.
                ;
000410  010146  4$:     MOV     R1,-(SP)        ;Save current line number
000412  010301         MOV     R3,R1           ;Get new virtual line number
000414  005000         CLR     R0              ;No secondary start-up command file
000416                 OCALL   INITLN          ;Initialize the line
000424  012601         MOV     (SP)+,R1        ;Get back original line number
000426  103414         BCS     20$             ;Br if unable to initiate new line
```

```
     58 000430  016102  000000G                 MOV     LNPRIM(R1),R2   ;Get primary line number
     59 000434  010263  000000G                 MOV     R2,LPARNT(R3)   ;Say primary job is parent of job started
     60 000440  016204  000000G                 MOV     LSECPT(R2),R4   ;Point to secondary line # table
     61 000444  060504                          ADD     R5,R4           ;Point to desired entry
     62 000446  110344                          MOVB    R3,-(R4)        ;Associate virtual line with primary
     63 000450  010263  000000G                 MOV     R2,LNPRIM(R3)   ;Set primary line number for virtual line
     64                                 ;
     65                                 ;  We are switching to a virtual line that has been initialized
     66                                 ;
     67 000454  004737  000470'         10$:    CALL    LINSWT          ;Set up line number prompt
     68                                 ;
     69                                 ;  Finished
     70                                 ;
     71 000460  012604                  20$:    MOV     (SP)+,R4
     72 000462  012603                          MOV     (SP)+,R3
     73 000464  012602                          MOV     (SP)+,R2
     74 000466  000207                          RETURN
```

```
                                        .SBTTL  LINSWT -- Switch to a virtual line
                                ;-----------------------------------------------
                                ;  LINSWT is called to switch association between a
                                ;  physical line and one of the virtual lines
                                ;  associated with the user.
                                ;
                                ;  Inputs:
                                ;   R1 = Current virtual line index number
                                ;   R5 = Number of the virtual to be switched to (0-MAXSEC)
                                ;
000470  010146                 LINSWT: MOV     R1,-(SP)
000472  010246                         MOV     R2,-(SP)
000474  010346                         MOV     R3,-(SP)
000476  010446                         MOV     R4,-(SP)
000500  010546                         MOV     R5,-(SP)
                                ;
                                ;  Set flag saying job no longer connected to terminal
                                ;
000502  052761  000000G 000000G         BIS     #$VNOTT,LSW(R1) ;Say not connected to terminal
                                ;
                                ;  Reduce execution priority of job we are disconnecting from
                                ;
000510  004737  000776'                 CALL    SPPRED          ;Reduce execution priority of discon proc
                                ;
                                ;  Clear flag that says bell has been rung for virtual line wait
                                ;  condition for line we are switching away from.
                                ;
000514  042761  000000G 000000G         BIC     #$VBELL,LSW9(R1);Say we need to ring bell for wait condition
                                ;
                                ;  If job we are switching away from has an active display window,
                                ;  tell the display window manager.
                                ;
000522  016102  000000G                 MOV     LWINDO(R1),R2   ;Does this job have an active window?
000526  001403                          BEQ     10$             ;Br if not
000530                                  OCALL   WINSF           ;Say we are switching from this job
                                ;
                                ;  Set info in virtual line tables
                                ;
000536  016102  000000G         10$:    MOV     LNPRIM(R1),R2   ;GET PRIMARY LINE #
000542  010201                          MOV     R2,R1
000544  005705                          TST     R5              ;DOES HE WANT PRIMARY LINE?
000546  001404                          BEQ     1$              ;BRANCH IF YES
000550  016201  000000G                 MOV     LSECPT(R2),R1   ;POINT TO SEC LINE # TABLE
000554  060501                          ADD     R5,R1           ;POINT TO DISIRED ENTRY
000556  114101                          MOVB    -(R1),R1        ;GET VIRTUAL LINE #
000560  010162  000000G         1$:     MOV     R1,LNMAP(R2)    ;SWITCH LINE CONTROL TO VIRTUAL LINE
000564  032761  000000G 000000G         BIT     #$INIT,LSW(R1)  ;HAS THIS LINE BEEN INITIALIZED?
000572  001470                          BEQ     6$              ;BR IF NOT
                                ;
                                ;  If the line we are switching to has a display window active,
                                ;  call window manager to let it know job has reconnected to terminal.
                                ;
000574  016102  000000G                 MOV     LWINDO(R1),R2   ;Display window active for new line?
000600  001404                          BEQ     8$              ;Br if not
000602                                  OCALL   WINST           ;Tell window manager about reconnect
000610  000447                          BR      9$
                                ;
```

```
                                ;  The line we are switching to does not have a display window active.
                                ;  At this point R1 contains the new line index #.
                                ;  Try to insert CR-LF-#->-CR-LF at head of
                                ;  new line's output buffer.
                                ;
000612                          8$:     DISABL                          ;** DISABLE **
000620  012704  000006                  MOV     #6,R4                   ;# OF CHARS WE WANT TO INSERT
000624  026104  000000G                 CMP     LOTSPC(R1),R4           ;ROOM FOR OUT CHARS?
000630  002434                          BLT     3$                      ;IF NOT THEN SKIP INSERT
000632  160461  000000G                 SUB     R4,LOTSPC(R1)           ;CLAIM THE SPACE
000636  012703  000770'                 MOV     #NLMSG,R3               ;POINT TO THE MESSAGE
000642  062705  000060                  ADD     #'0,R5                  ;CONVERT VIRTUAL # TO ASCII CHAR
000646  110537  000773'                 MOVB    R5,<NLMSG+3>            ;MOVE INTO MESSAGE
000652  016102  000000G                 MOV     LOTPNT(R1),R2           ;POINT TO HEAD OF OUTPUT LIST
000656                                  TTMAP                           ;MAP TO TT BUFFER AREA
000672  020261  000000G         5$:     CMP     R2,LOTBUF(R1)           ;AT HEAD OF BUFFER?
000676  101002                          BHI     4$                      ;BRANCH IF NOT
000700  016102  000000G                 MOV     LOTEND(R1),R2           ;WRAP AROUND TO END
000704  112342                  4$:     MOVB    (R3)+,-(R2)             ;MOVE MESSAGE TO USER'S BUFFER
000706  077407                          SOB     R4,5$                   ;LOOP IF MORE TO DO
000710  010261  000000G                 MOV     R2,LOTPNT(R1)           ;UPDATE CHARACTER POINTER
000714                                  UNMAP                           ;RESTORE MAPPING
000722                          3$:     ENABL                           ;** ENABLE **
                                ;
                                ;  Set up info for job we are switching to
                                ;
000730  042761  000000G 000000G 9$:     BIC     #$VNOTT,LSW(R1) ;SAY RECONNECTED TO TERMINAL
000736  116161  000000G 000000G         MOVB    LBSPRI(R1),LPRI(R1);SET CURRENT PRI = BASE PRI
000744  004777  000000G                 CALL    @TRNSTR         ;TRY TO START PRINTING
000750  004737  000000G                 CALL    FORCEX          ;FORCE EXECUTION OF NEW LINE
                                ;
                                ;  Finished
                                ;
000754  012605                  6$:     MOV     (SP)+,R5
000756  012604                          MOV     (SP)+,R4
000760  012603                          MOV     (SP)+,R3
000762  012602                          MOV     (SP)+,R2
000764  012601                          MOV     (SP)+,R1
000766  000207                          RETURN
                                ;
000770    012     015     076   NLMSG:  .ASCII  <12><15>/>#/<12><15>
000773    043     012     015
                                        .EVEN
```

```
                                        .SBTTL  SPPRED -- Reduce execution priority for disconnect process
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to reduce the execution priority of a subprocess
                                ;  as it disconnects from terminal control.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index of process whose priority is to be reduced
                                ;
000776  010246                  SPPRED: MOV     R2,-(SP)
001000  116102  0000000                 MOVB    LBSPRI(R1),R2   ;GET BASE PRIORITY FOR JOB
001004  120237  0000000                 CMPB    R2,VPRILO       ;IS THIS A FIXED PRIORITY JOB?
001010  101414                          BLOS    7$              ;BR IF YES
001012  120237  0000000                 CMPB    R2,VPRIHI       ;REAL-TIME JOB?
001016  103011                          BHIS    7$              ;BR IF YES
001020  113700  0000000                 MOVB    VPRIVR,R0       ;GET AMT TO REDUCE FOR DISCONNECTED JOBS
001024  160002                          SUB     R0,R2           ;REDUCE FOR DISCONNECTED JOB
001026  120237  0000000                 CMPB    R2,VPRILO       ;DID PRIORITY GO OUT OF NORMAL RANGE?
001032  003003                          BGT     7$              ;BR IF NOT
001034  113702  0000000                 MOVB    VPRILO,R2       ;GET LOWEST NORMAL PRIORITY
001040  005202                          INC     R2
001042  110261  0000000          7$:     MOVB    R2,LPRI(R1)     ;SET CURRENT PRIORITY FOR JOB
                                ;
                                ;  Finished
                                ;
001046  012602                          MOV     (SP)+,R2
001050  000207                          RETURN
```

```
                        .SBTTL  STTCPL -- Set completion routine for TT activation
                ;-----------------------------------------------------------------------
                ;  Process the EMT that connects a completion routine to TT input
                ;  character reception.
                ;
                ;  EMT argument block:
                ;
                ;       .BYTE   1,133
                ;       .WORD   completion_routine
                ;
                ;  Inputs:
                ;   R1 = Job index number
                ;
001052          STTCPL:
                ;
                ;  Cancel any existing completion routine connection.
                ;
001052  010102          MOV     R1,R2           ;Save job index number in R2
001054  005037  000000G  CLR     @R0             ;Assume no completion routine now
001060  016201  000000G  MOV     LTTCR(R2),R1    ;Get addr of completion Q element
001064  001407          BEQ     1$              ;Br if none
001066  016137  000000G 000000G  MOV     CQ$RTN(R1),@R0  ;Return old compl rtn address in R0
001074  005062  000000G  CLR     LTTCR(R2)       ;Say no completion Q element now
001100  004737  000000G  CALL    QFREE           ;Release the queue element
                ;
                ;  See if the address of a new completion routine was specified
                ;
001104  013700  000020G 1$:     MOV     EMTBLK+2,R0     ;Did he specify a compl rtn address?
001110  001442          BEQ     9$              ;Br if not
001112  004737  000000G  CALL    VALADW          ;Make sure compl routine addr is valid
                ;
                ;  Get a queue element and set up as a completion queue element
                ;
001116  004737  000000G  CALL    GETQ            ;Get a queue element (address in R1)
001122  010104          MOV     R1,R4           ;Get address of queue element
001124  110264  000000G  MOVB    R2,CQ$JOB(R4)   ;Set job number in Q element
001130  112764  000000G 000000G  MOVB    #S$TTSC,CQ$RNS(R4);Set execution state for job
001136  032762  000000G 000000G  BIT     #$NOINT,LSW7(R2);Is job running non-interactively?
001144  001403          BEQ     2$              ;Br if not
001146  112764  000000G 000000G  MOVB    #S$IOFN,CQ$RNS(R4);Set non-interactive execution state
001154  116264  000000G 000000G 2$:     MOVB    LPRI(R2),CQ$PRI(R4);Set execution priority
001162  112764  000000G 000000G  MOVB    #CP$STD,CQ$CP(R4);Set compl routine class priority
001170  013764  000020G 000000G  MOV     EMTBLK+2,CQ$RTN(R4);Set address of compl routine
001176  013764  000000G 000000G  MOV     EMTMAP,CQ$PA5(R4);Set EMT entry mapping for par 5
001204  010462  000000G  MOV     R4,LTTCR(R2)    ;Set address of queue element
                ;
                ;  If there are any characters pending now, trigger a completion
                ;  routine immediately.
                ;
001210  010201          MOV     R2,R1           ;Get job index # to R1 for TTCPL
001212  004737  001222'  CALL    TTCPL           ;See if we need to do compl rtn now
                ;
                ;  Finished
                ;
001216  000137  000000G 9$:     JMP     EMTXIT
```

```
                                        .SBTTL  TTCPL  -- Queue a TT completion routine
                                ;-----------------------------------------------------------------------
                                ;  Check to see if there are any input characters pending and the job
                                ;  wants a completion routine executed for input characters.
                                ;  If so, queue the completion request.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
001222  010246                  TTCPL:  MOV     R2,-(SP)
001224  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if job want completion for character input
                                ;
001226                                  DISABL                  ;;;** Disable interrupts **
001234  016104  0000000                 MOV     LTTCR(R1),R4    ;;;Does job want TT input compl routine?
001240  001434                          BEQ     9$              ;;;Br if not
001242  016102  0000000                 MOV     LINPNT(R1),R2   ;;;Get pointer to next char to get
001246  020261  0000000                 CMP     R2,LINNXT(R1)   ;;;Are there any pending input chars?
001252  001427                          BEQ     9$              ;;;Br if not
001254  005061  0000000                 CLR     LTTCR(R1)       ;;;Say we are using completion routine
001260                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Get next pending input character
                                ;
001266                                  OCALL   DELCHR          ;Get next char from input buffer
001274  103414                          BCS     8$              ;Br if no chars available
                                ;
                                ;  We got a character
                                ;
001276  005061  0000000                 CLR     LRTCHR(R1)      ;Say no read time-out pending
001302  042700  177400                  BIC     #^C<377>,R0     ;Clear all but low-order 8 bits of char
001306                                  OCALL   CVTLC           ;Possibly convert to upper case
                                ;
                                ;  Queue the completion routine
                                ;
001314  010064  0000000                 MOV     R0,CQ$R0(R4)    ;Pass char to compl rtn in R0
001320  004737  0000000                 CALL    QCOMPL          ;Queue the completion routine
001324  000402                          BR      9$
                                ;
                                ;  No characters available.
                                ;  Leave the completion request pending.
                                ;
001326  010461  0000000         8$:     MOV     R4,LTTCR(R1)    ;Say compl request still pending
                                ;
                                ;  Finished
                                ;
001332                          9$:     ENABL                   ;** Make sure interrupts are enabled **
001340  012604                          MOV     (SP)+,R4
001342  012602                          MOV     (SP)+,R2
001344  000207                          RETURN
                                ;
000001                                  .END
Errors detected:   0

*** Assembler statistics
```

```
Work  file  reads:  0
Work  file writes:  0
Size of work file:  248 Words  ( 1 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:13.42
DK:TSTTY2,LP:TSTTY2=DK:TSTTY2.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

$INIT	1-31	5-47				
$LOFCF	1-28	4-17				
$NABRS	1-27	2-72	2-85			
$NOIN	1-28	4-19				
$NOINT	1-23	7-38				
$NOVLN	1-28	4-15				
$RBRK	1-27	2-28	2-90			
$SUCF	1-28	4-17				
$VBELL	1-24	5-28				
$VNOTT	1-29	5-19	5-84			
ASTBL1	2-36	3-4#				
ASTBL2	2-42	3-21#				
AUTSPD	1-18	2-14#				
CP$STD	1-25	7-42				
CQ$CP	1-26	7-42*				
CQ$JOB	1-25	7-36*				
CQ$PA5	1-22	7-44*				
CQ$PRI	1-25	7-41*				
CQ$RO	1-23	8-37*				
CQ$RNS	1-25	7-37*	7-40*			
CQ$RTN	1-26	7-22	7-43*			
CVTLC	1-23	8-33				
DELCHR	1-22	8-26				
DOSWIT	1-18	4-9#				
EMTBLK	1-24	7-28	7-43			
EMTMAP	1-22	7-44				
EMTXIT	1-24	7-55				
FORCEX	1-32	5-87				
FSTSL	1-29	4-40				
GETQ	1-22	7-34				
INITLN	1-29	4-55				
INTPRI	1-32	5-80	8-22	8-48		
KPAR6	1-32	5-72	5-72*	5-79*		
LABTIM	1-27	2-22	2-71*	2-84*		
LBSPRI	1-30	5-85	6-10			
LINNXT	1-26	8-19				
LINPNT	1-22	8-18				
LINSWT	1-18	4-67	5-11#			
LMXPRM	1-27	2-37				
LNMAP	1-31	5-46*				
LNPRIM	1-28	4-30	4-43	4-58	4-63*	5-39
LOTBUF	1-31	5-73				
LOTEND	1-31	5-75				
LOTPNT	1-31	5-71	5-78*			
LOTSPC	1-31	5-65	5-67*			
LPARNT	1-29	4-59*				
LPRI	1-25	1-30	5-85*	6-21*	7-41	
LRTCHR	1-23	8-31*				
LSECPT	1-29	1-29	4-31	4-60	5-43	
LSTSL	1-29	4-41				
LSW	1-29	5-19*	5-47	5-84*		
LSW10	1-27	2-28	2-90*			
LSW2	1-28	4-15				
LSW3	1-28	4-19				
LSW7	1-23	7-38				
LSW9	1-28	2-72*	2-85*	4-17	5-28*	

Cross reference table (CREF V05.04)

LTTCR	1-24	7-20	7-23*	7-45*	8-16	8-21*	8-44*
LTTPAR	1-32	5-72					
LWINDO	1-30	5-33	5-53				
MAPHLD	1-86#	5-72*	5-79				
NLMSG	5-68	5-70*	5-98#				
OVRHC	1-32	4-55	5-35	5-55	8-26	8-33	
PR7	1-32	5-63	8-15				
PSW	1-32	5-63*	5-80*	8-15*	8-22*	8-48*	
QCOMPL	1-26	8-38					
QFREE	1-22	7-24					
S$IOFN	1-23	7-40					
S$TTSC	1-22	7-37					
S110	1-33	3-30	3-31				
S1200	1-33	3-16	3-22	3-23	3-24	3-25	3-26
S150	1-33						
S1800	1-33						
S19200	1-33	3-4	3-5	3-6	3-7	3-8	3-9
S2400	1-33	3-14	3-21				
S300	1-33	3-28					
S3600	1-33						
S4800	1-33	3-11					
S600	1-27	2-30	2-43				
S9600	1-27	2-40	2-65	3-10			
SETSPD	1-27	2-31	2-66	2-79			
SPPRED	1-18	5-23	6-9#				
STTCPL	1-18	7-14#					
TRNSTR	1-32	5-86					
TSTTY2	1-6#						
TTCPL	1-18	7-51	8-10#				
URO	1-22	7-19*	7-22*				
VALADW	1-24	7-30					
VPRIHI	1-30	6-13					
VPRILO	1-30	6-11	6-17	6-19			
VPRIVR	1-30	6-15					
WINSF	1-30	5-35					
WINST	1-31	5-55					

Cross reference table (CREF V05.04)

DISABL	1-37#	5-63	8-15			
ENABL	1-41#	5-80	8-22	8-48		
OCALL	1-45#	4-55	5-35	5-55	8-26	8-33
TTMAP	1-75#	5-72				
TTMAPX	1-66#					
UNMAP	1-80#	5-79				
UNMAPX	1-72#					