

Table of contents

2-	1	AUTSPD -- Do autobaud speed selection for a line
4-	1	DOSWIT -- Switch to a virtual line
5-	1	LINSWT -- Switch to a virtual line
6-	1	SPPRED -- Reduce execution priority for disconnect process
7-	1	STTCPL -- Set completion routine for TT activation
8-	1	TTCPL -- Queue a TT completion routine

```

1          .TITLE  TSTTY2 -- Auxiliary terminal control overlay
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  QBL
5 000000    .CSECT  TSTTY2
6 000000    100100    TSTTY2: .RAD50  /TT2/          ;Overlay region id
7
8          ;
9          ; TSTTY2 is an auxiliary terminal control overlay.
10         ; The routines in TSTTY2 could logically go in TSTTY except
11         ; that overlay has already reached the 8Kb size limit.
12
13         ; Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985.
14         ; S&H Computer Systems, Inc.
15         ; Nashville, Tennessee USA
16
17         ; Global definitions
18         ;
19         ; .GLOBL  AUTSPD, DOSWIT, LINSWT, STTCPL, TTCPL, SPPRED
20
21         ; Global references
22         ;
23         ; .GLOBL  URO, QFREE, GETQ, S$TTSC, EMTMAP, CQ$PA5, LINPNT, DELCHR
24         ; .GLOBL  LRTCHR, CVTLC, CQ$RO, S$IOFN, $NOINT, LSW7
25         ; .GLOBL  $VBELL, EMTBLK, VALADW, LTTCR, EMTXIT
26         ; .GLOBL  CQ$JOB, CQ$RNS, LPRI, CQ$PRI, CP$STD
27         ; .GLOBL  CQ$CP, CQ$RTN, QCOMPL, LINNXT
28         ; .GLOBL  LABTIM, $RBRK, LSW10, S600, SETSPD, LMXPRM, S9600, $NABRS
29         ; .GLOBL  LSW9, $NOVLN, LSW2, $SUCF, $LOFCF, $NOIN, LSW3, LNPRIM
30         ; .GLOBL  LSECPT, FSTSL, LSTSL, INITLN, LPARNT, LSECPT, $VNQTT, LSW
31         ; .GLOBL  LBSPRI, VPRILO, VPRIHI, VPRIVR, LPRI, LWINDO, WINSF
32         ; .GLOBL  LNMAP, $INIT, WINST, LOTSPC, LOTPNT, LOTBUF, LOTEND
33         ; .GLOBL  TRNSTR, FORCEX, OVRHC, PR7, PSW, KPAR6, LTTPAR, INTPRI
34         ; .GLOBL  S19200, S4800, S3600, S2400, S1800, S1200, S300, S150, S110
35
36         ; Macro definitions
37         ;
38         ; .MACRO  DISABL          ;DISABLE INTERRUPTS
39         BIS      #PR7, @#PSW
40         .ENDM    DISABL
41
42         ; .MACRO  ENABL          ;ENABLE INTERRUPTS
43         BIC      INTPRI, @#PSW
44         .ENDM    ENABL
45
46         ; .MACRO  OCALL  ENTADD
47         .IF      B, ENTADD
48         .ERROR   ;OCALL without entry address
49         .ENDC
50         CALL     OVRHC
51         .WORD    ENTADD
52         .ENDM    OCALL
53
54         ; The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
55         ; terminal character buffer area. The previous contents of par6 map
56         ; register are pushed on the stack and may be restored by using the
57         ; UNMAP or UNMAPX macros.
58         ; R1 must contain the line index number of the line whose buffers

```

```

58      ; are being accessed.
59      ; The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
60      ; macros is that the X-versions are more efficient but may only be
61      ; used from within interrupt service routines where we are guaranteed
62      ; to be running on the system stack.
63      ; The TTMAP and UNMAP versions of the macros must only be
64      ; used in sections of code where the interrupts are disabled.
65      ;
66      .MACRO  TTMAPX
67      MOV     LTPAR(R1),@#KPAR6
68      .ENDM   TTMAPX
69
70      ;
71      ;
72      .MACRO  UNMAPX
73      .ENDM   UNMAPX
74
75      .MACRO  TTMAP
76      MOV     @KPAR6,MAPHLD
77      MOV     LTPAR(R1),@#KPAR6
78      .ENDM   TTMAP
79
80      .MACRO  UNMAP
81      MOV     MAPHLD,@#KPAR6
82      .ENDM   UNMAP
83
84      ; Data areas
85      ;
86 000002 000000  MAPHLD: .WORD 0 ; TEMP CELL USED BY TTMAP MACRO

```

AUTSPD -- Do autobaud speed selection for a line

```

1          .SBTTL  AUTSPD -- Do autobaud speed selection for a line
2          ;-----
3          ; AUTSPD is called when a character is received for a line that is not
4          ; logged on and which is to have autobaud speed selection.
5          ;
6          ; Inputs:
7          ;   R4 = Line index number.
8          ;   R5 = Character received.
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> Start this line now.
12         ;   C-flag set    ==> Do not start this line.
13         ;
14 000004 010146  AUTSPD: MOV     R1,-(SP)
15 000006 010246      MOV     R2,-(SP)
16 000010 010346      MOV     R3,-(SP)
17 000012 010401      MOV     R4,R1      ; Carry line index number in R1
18 000014 042705 177600      BIC     #^C177,R5      ; Mask character to 7 bits
19         ;
20         ; Ignore characters that come in too fast
21         ;
22 000020 026127 0000000 000142      CMP     LABTIM(R1),#98. ; Should we check this character?
23 000026 101055      BHI     10$      ; Br if not
24         ;
25         ; If we received a break (framing error), set the speed to 600 baud
26         ; and get another character for autobaud detection.
27         ;
28 000030 032761 0000000 0000000      BIT     #$RBRK,LSW10(R1); Did we receive a break?
29 000036 001405      BEQ     6$      ; Br if not
30 000040 012700 0000000      MOV     #S600,R0      ; Select 600 baud
31 000044 004737 0000000      CALL    SETSPD      ; Set line speed
32 000050 000436      BR      8$      ; Don't start line yet
33         ;
34         ; See if we should search speed table 1 or 2
35         ;
36 000052 012702 000230' 6$:      MOV     #ASTBL1,R2      ; Assume we are running at 9600 baud
37 000056 016103 0000000      MOV     LMXPRM(R1),R3      ; Get current line speed
38 000062 000303      SWAB     R3      ; Get to low-order byte
39 000064 042703 177760      BIC     #^C17,R3      ; Clear all but speed code
40 000070 020327 0000000      CMP     R3,#S9600      ; Is speed currently 9600?
41 000074 001411      BEQ     1$      ; Br if yes
42 000076 012702 000256'      MOV     #ASTBL2,R2      ; Assume speed is 600 baud
43 000102 020327 0000000      CMP     R3,#S600      ; Is speed currently 600?
44 000106 001404      BEQ     1$      ; Br if yes
45         ;
46         ; Current speed is neither 9600 baud nor 600 baud.
47         ; If character is carriage return, assume current speed is correct.
48         ;
49 000110 120527 000015      CMPB     R5,#15      ; Is character carriage return?
50 000114 001427      BEQ     11$      ; Br if yes -- speed must be correct
51 000116 000407      BR      5$      ; Set speed to 9600 baud
52         ;
53         ; See if we can find the received character in the autobaud table
54         ;
55 000120 105762 000001 1$:      TSTB     1(R2)      ; Have we reached the end of the table?
56 000124 001404      BEQ     5$      ; Br if at end of table
57 000126 120522      CMPB     R5,(R2)+      ; Does received char match this table entry?

```

AUTSPD -- Do autobaud speed selection for a line

```

58 000130 001416          BEQ      2$          ;Br if yes -- We found the speed
59 000132 105722          TSTB     (R2)+       ;Skip over speed code for this entry
60 000134 000771          BR       1$          ;Loop if more entries to check
61                          ;
62                          ; We did not find a matching character in this table.
63                          ; Reset speed to 9600 baud.
64                          ;
65 000136 012700 0000000 5$:      MOV      #9600,R0      ;Select 9600 baud
66 000142 004737 0000000      CALL     SETSPD          ;Set the line speed
67                          ;
68                          ; We could not identify this character.
69                          ; Start timer which will reset the baud rate to 9600 after 10 seconds.
70                          ;
71 000146 012761 000144 0000000 8$:      MOV      #100.,LABTIM(R1);Start autobaud timer for this line
72 000154 052761 0000000 0000000      BIS      #$NABRS,LSW7(R1);Reset speed to 9600 baud after time interval
73 000162 000261          10$:      SEC                      ;Say not to start the line this time
74 000164 000412          BR       9$          ;Finished
75                          ;
76                          ; We found the correct speed entry for this line
77                          ;
78 000166 112200          2$:      MOVB     (R2)+,R0      ;Get the correct speed code
79 000170 004737 0000000      CALL     SETSPD          ;Set the line speed
80                          ;
81                          ; Set a short timer to cause us to ignore any garbage characters
82                          ; that may be received after the autobaud character.
83                          ;
84 000174 012761 000002 0000000 11$:     MOV      #2.,LABTIM(R1) ;Ignore input chars for 0.2 seconds
85 000202 042761 0000000 0000000      BIC      #$NABRS,LSW7(R1);Clear need-reset flag
86 000210 000241          CLC                      ;Signal to start the line
87                          ;
88                          ; Finished
89                          ;
90 000212 042761 0000000 0000000 9$:     BIC      #$RBRK,LSW10(R1);Clear break-received flag for line
91 000220 012603          MOV      (SP)+,R3
92 000222 012602          MOV      (SP)+,R2
93 000224 012601          MOV      (SP)+,R1
94 000226 000207          RETURN

```

```
1 ;
2 ; Table to indicate speed when we are receiving at 9600 baud
3 ;
4 000230      177      0000      ASTBL1: . BYTE      177, S19200
5 000232      176      0000              . BYTE      176, S19200
6 000234      175      0000              . BYTE      175, S19200
7 000236      172      0000              . BYTE      172, S19200
8 000240      171      0000              . BYTE      171, S19200
9 000242      162      0000              . BYTE      162, S19200
10 000244      015      0000              . BYTE      015, S9600
11 000246      146      0000              . BYTE      146, S4800
12 ; ; . BYTE      014, S3600
13 ; ; . BYTE      176, S2400
14 000250      170      0000              . BYTE      170, S2400
15 ; ; . BYTE      160, S1800
16 000252      000      0000              . BYTE      000, S1200
17 000254      000      000              . BYTE      0, 0
18 ;
19 ; Table to indicate speed when we are receiving at 600 baud
20 ;
21 000256      177      0000      ASTBL2: . BYTE      177, S2400
22 000260      176      0000              . BYTE      176, S1200
23 000262      172      0000              . BYTE      172, S1200
24 000264      171      0000              . BYTE      171, S1200
25 000266      170      0000              . BYTE      170, S1200
26 000270      162      0000              . BYTE      162, S1200
27 ; ; . BYTE      015, S600
28 000272      146      0000              . BYTE      146, S300
29 ; ; . BYTE      170, S150
30 000274      160      0000              . BYTE      160, S110
31 000276      140      0000              . BYTE      140, S110
32 000300      000      000              . BYTE      0, 0
```

```

1          .SBTTL  DOSWIT -- Switch to a virtual line
2          ;-----
3          ; DOSWIT is called when we receive a control-W-digit sequence.
4          ;
5          ; Inputs:
6          ;   R1 = Current virtual line index number.
7          ;   R5 = Binary value of virtual line selected.
8          ;
9 000302    010246    DOSWIT: MOV     R2,-(SP)
10 000304    010346      MOV     R3,-(SP)
11 000306    010446      MOV     R4,-(SP)
12          ;
13          ; See if this job is allowed to use virtual lines
14          ;
15 000310    032761    0000000 0000000      BIT     #$NOVLN,LSW2(R1);Can job use virtual lines?
16 000316    001060      BNE     20$          ;Br if not
17 000320    032761    0000000 0000000      BIT     #<$SUCF!$LOFCF>,LSW9(R1);Doing logon/logoff processing?
18 000326    001054      BNE     20$          ;Br if yes
19 000330    032761    0000000 0000000      BIT     #$NOIN,LSW3(R1);Doing logon/logoff processing?
20 000336    001050      BNE     20$          ;Br if yes -- don't allow line switching
21          ;
22          ; Determine if we are switching to a virtual line or back to the primary
23          ;
24 000340    005705      TST     R5          ;Switching to primary line?
25 000342    001444      BEQ     10$          ;Br if switching to primary line
26          ;
27          ; We are switching to a virtual line
28          ; See if line has been allocated yet.
29          ;
30 000344    016102    0000000      MOV     LNPRIM(R1),R2 ;Get primary line #
31 000350    016203    0000000      MOV     LSECPT(R2),R3 ;Point to table of secondary line numbers
32 000354    060503      ADD     R5,R3        ;Point to entry for selected line
33 000356    105743      TSTB    -(R3)        ;Has this virtual line been allocated yet?
34 000360    001035      BNE     10$          ;Br if yes
35          ;
36          ; Request to switch to a virtual line which has not been
37          ; allocated yet.
38          ; See if we can find a free virtual line.
39          ;
40 000362    012703    0000000      MOV     #FSTSL,R3    ;Get # of first virtual line
41 000366    020327    0000000 3$:  CMP     R3,#LSTSL    ;Checked all?
42 000372    101032      BHI     20$          ;Br if yes
43 000374    005763    0000000      TST     LNPRIM(R3)    ;Is this line available?
44 000400    001403      BEQ     4$          ;Br if yes
45 000402    062703    0000002      ADD     #2,R3        ;Get index to next virtual line
46 000406    000767      BR      3$
47          ;
48          ; We have found a free virtual line.
49          ; It's index is in R3.
50          ; Initialize it.
51          ;
52 000410    010146    4$:  MOV     R1,-(SP)    ;Save current line number
53 000412    010301      MOV     R3,R1        ;Get new virtual line number
54 000414    005000      CLR     R0          ;No secondary start-up command file
55 000416      OCALL    INITLN              ;Initialize the line
56 000424    012601      MOV     (SP)+,R1    ;Get back original line number
57 000426    103414      BCS     20$          ;Br if unable to initiate new line

```

```
58 000430 016102 0000000 MOV LNPRIM(R1),R2 ;Get primary line number
59 000434 010263 0000000 MOV R2,LPARNT(R3) ;Say primary job is parent of job started
60 000440 016204 0000000 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 0000000 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
```

LINSWT -- Switch to a virtual line

```

1          .SBTTL  LINSWT -- Switch to a virtual line
2          ;-----
3          ; LINSWT is called to switch association between a
4          ; physical line and one of the virtual lines
5          ; associated with the user.
6          ;
7          ; Inputs:
8          ;   R1 = Current virtual line index number
9          ;   R5 = Number of the virtual to be switched to (0-MAXSEC)
10         ;
11 000470 010146 LINSWT: MOV      R1,-(SP)
12 000472 010246      MOV      R2,-(SP)
13 000474 010346      MOV      R3,-(SP)
14 000476 010446      MOV      R4,-(SP)
15 000500 010546      MOV      R5,-(SP)
16         ;
17         ; Set flag saying job no longer connected to terminal
18         ;
19 000502 052761 0000000 0000000      BIS      #$VNOTT,LSW(R1) ; Say not connected to terminal
20         ;
21         ; Reduce execution priority of job we are disconnecting from
22         ;
23 000510 004737 000776'      CALL      SPPRED      ; Reduce execution priority of discon proc
24         ;
25         ; Clear flag that says bell has been rung for virtual line wait
26         ; condition for line we are switching away from.
27         ;
28 000514 042761 0000000 0000000      BIC      #$VBELL,LSW(R1); Say we need to ring bell for wait condition
29         ;
30         ; If job we are switching away from has an active display window,
31         ; tell the display window manager.
32         ;
33 000522 016102 0000000      MOV      LWINDO(R1),R2 ; Does this job have an active window?
34 000526 001403      BEQ      10$      ; Br if not
35 000530      OCALL      WINSF      ; Say we are switching from this job
36         ;
37         ; Set info in virtual line tables
38         ;
39 000536 016102 0000000      10$: MOV      LNPRIM(R1),R2 ; GET PRIMARY LINE #
40 000542 010201      MOV      R2,R1
41 000544 005705      TST      R5      ; DOES HE WANT PRIMARY LINE?
42 000546 001404      BEQ      1$      ; BRANCH IF YES
43 000550 016201 0000000      MOV      LSECPT(R2),R1 ; POINT TO SEC LINE # TABLE
44 000554 060501      ADD      R5,R1      ; POINT TO DISIRED ENTRY
45 000556 114101      MOVB     -(R1),R1      ; GET VIRTUAL LINE #
46 000560 010162 0000000      1$: MOV      R1,LNMAP(R2) ; SWITCH LINE CONTROL TO VIRTUAL LINE
47 000564 032761 0000000 0000000      BIT      #$INIT,LSW(R1) ; HAS THIS LINE BEEN INITIALIZED?
48 000572 001470      BEQ      6$      ; BR IF NOT
49         ;
50         ; If the line we are switching to has a display window active,
51         ; call window manager to let it know job has reconnected to terminal.
52         ;
53 000574 016102 0000000      MOV      LWINDO(R1),R2 ; Display window active for new line?
54 000600 001404      BEQ      8$      ; Br if not
55 000602      OCALL      WINST      ; Tell window manager about reconnect
56 000610 000447      BR      9$
57         ;

```

LINSWT -- Switch to a virtual line

```

58      ; The line we are switching to does not have a display window active.
59      ; At this point R1 contains the new line index #.
60      ; Try to insert CR-LF-#->-CR-LF at head of
61      ; new line's output buffer.
62      ;
63 000612      8$:  DISABL      ; ** DISABLE **
64 000620      012704 000006      MOV      #6,R4      ; # OF CHARS WE WANT TO INSERT
65 000624      026104 0000000     CMP      LOTSPC(R1),R4 ; ROOM FOR OUT CHARS?
66 000630      002434          BLT      3$      ; IF NOT THEN SKIP INSERT
67 000632      160461 0000000     SUB      R4,LOTSPC(R1) ; CLAIM THE SPACE
68 000636      012703 000770'     MOV      #NLMSG,R3      ; POINT TO THE MESSAGE
69 000642      062705 000060      ADD      #'0,R5      ; CONVERT VIRTUAL # TO ASCII CHAR
70 000646      110537 000773'     MOVB     R5,<NLMSG+3>    ; MOVE INTO MESSAGE
71 000652      016102 0000000     MOV      LOTPNT(R1),R2 ; POINT TO HEAD OF OUTPUT LIST
72 000656          TTMAP          ; MAP TO TT BUFFER AREA
73 000672      020261 0000000     5$:  CMP      R2,LOTBUF(R1) ; AT HEAD OF BUFFER?
74 000676      101002          BHI      4$      ; BRANCH IF NOT
75 000700      016102 0000000     MOV      LOTEND(R1),R2 ; WRAP AROUND TO END
76 000704      112342          4$:  MOVB     (R3)+,-(R2) ; MOVE MESSAGE TO USER'S BUFFER
77 000706      077407          SDB      R4,5$      ; LOOP IF MORE TO DO
78 000710      010261 0000000     MOV      R2,LOTPNT(R1) ; UPDATE CHARACTER POINTER
79 000714          UNMAP          ; RESTORE MAPPING
80 000722      3$:  ENABL      ; ** ENABLE **
81      ;
82      ; Set up info for job we are switching to
83      ;
84 000730      042761 0000000 0000000 9$:  BIC      #$VNOTT,LSW(R1) ; SAY RECONNECTED TO TERMINAL
85 000736      116161 0000000 0000000     MOVB     LBSPRI(R1),LPRI(R1) ; SET CURRENT PRI = BASE PRI
86 000744      004777 0000000          CALL     @TRNSTR      ; TRY TO START PRINTING
87 000750      004737 0000000          CALL     FORCEX      ; FORCE EXECUTION OF NEW LINE
88      ;
89      ; Finished
90      ;
91 000754      012605          6$:  MOV      (SP)+,R5
92 000756      012604          MOV      (SP)+,R4
93 000760      012603          MOV      (SP)+,R3
94 000762      012602          MOV      (SP)+,R2
95 000764      012601          MOV      (SP)+,R1
96 000766      000207          RETURN
97      ;
98 000770      012      015      076      NLMSG:  . ASCII <12><15>/>#/<12><15>
99      000773      043      012      015          . EVEN

```

```

1          .SBTTL  SPPRED -- Reduce execution priority for disconnect process
2          ;-----
3          ; This routine is called to reduce the execution priority of a subprocess
4          ; as it disconnects from terminal control.
5          ;
6          ; Inputs:
7          ; R1 = Job index of process whose priority is to be reduced
8          ;
9 000776 010246      SPPRED: MOV      R2, -(SP)
10 001000 116102 0000000  MOV      R2, LBSPRI(R1), R2      ; GET BASE PRIORITY FOR JOB
11 001004 120237 0000000  CMP      R2, VPRILO      ; IS THIS A FIXED PRIORITY JOB?
12 001010 101414      BLOS      7$      ; BR IF YES
13 001012 120237 0000000  CMP      R2, VPRIHI      ; REAL-TIME JOB?
14 001016 103011      BHIS      7$      ; BR IF YES
15 001020 113700 0000000  MOV      R0, R2      ; GET AMT TO REDUCE FOR DISCONNECTED JOBS
16 001024 160002      SUB      R0, R2      ; REDUCE FOR DISCONNECTED JOB
17 001026 120237 0000000  CMP      R2, VPRILO      ; DID PRIORITY GO OUT OF NORMAL RANGE?
18 001032 003003      BGT      7$      ; BR IF NOT
19 001034 113702 0000000  MOV      R2, VPRILO, R2      ; GET LOWEST NORMAL PRIORITY
20 001040 005202      INC      R2
21 001042 110261 0000000  7$: MOV      R2, LPRI(R1)      ; SET CURRENT PRIORITY FOR JOB
22          ;
23          ; Finished
24          ;
25 001046 012602      MOV      (SP)+, R2
26 001050 000207      RETURN
  
```

```

1          .SBTTL  STTCPL -- Set completion routine for TT activation
2          ;-----
3          ; Process the EMT that connects a completion routine to TT input
4          ; character reception.
5          ;
6          ; EMT argument block:
7          ;
8          ;     .BYTE    1,133
9          ;     .WORD    completion_routine
10         ;
11         ; Inputs:
12         ;     R1 = Job index number
13         ;
14 001052   STTCPL:
15         ;
16         ; Cancel any existing completion routine connection.
17         ;
18 001052   010102   MOV     R1,R2           ; Save job index number in R2
19 001054   005037   CLR     URO             ; Assume no completion routine now
20 001060   016201   MOV     LTTCLR(R2),R1   ; Get addr of completion Q element
21 001064   001407   BEQ     1$              ; Br if none
22 001066   016137   MOV     CQ$RTN(R1),URO  ; Return old compl rtn address in R0
23 001074   005062   CLR     LTTCLR(R2)     ; Say no completion Q element now
24 001100   004737   CALL    QFREE          ; Release the queue element
25         ;
26         ; See if the address of a new completion routine was specified
27         ;
28 001104   013700   1$:  MOV     EMTBLK+2,R0 ; Did he specify a compl rtn address?
29 001110   001442   BEQ     9$              ; Br if not
30 001112   004737   CALL    VALADW         ; Make sure compl routine addr is valid
31         ;
32         ; Get a queue element and set up as a completion queue element
33         ;
34 001116   004737   CALL    GETQ           ; Get a queue element (address in R1)
35 001122   010104   MOV     R1,R4           ; Get address of queue element
36 001124   110264   MOVB    R2,CQ$JOB(R4)  ; Set job number in Q element
37 001130   112764   MOVB    #S$TTSC,CQ$RNS(R4); Set execution state for job
38 001136   032762   BIT     #NOINT,LSW7(R2); Is job running non-interactively?
39 001144   001403   BEQ     2$              ; Br if not
40 001146   112764   MOVB    #S$IOFN,CQ$RNS(R4); Set non-interactive execution state
41 001154   116264   2$:  MOVB    LPRI(R2),CQ$PRI(R4); Set execution priority
42 001162   112764   MOVB    #CP$STD,CQ$CP(R4); Set compl routine class priority
43 001170   013764   MOV     EMTBLK+2,CQ$RTN(R4); Set address of compl routine
44 001176   013764   MOV     EMTMAP,CQ$PA5(R4); Set EMT entry mapping for par 5
45 001204   010462   MOV     R4,LTTCLR(R2)  ; Set address of queue element
46         ;
47         ; If there are any characters pending now, trigger a completion
48         ; routine immediately.
49         ;
50 001210   010201   MOV     R2,R1           ; Get job index # to R1 for TTCPL
51 001212   004737   CALL    TTCPL          ; See if we need to do compl rtn now
52         ;
53         ; Finished
54         ;
55 001216   000137   9$:  JMP     EMTXIT

```

TTCPL -- Queue a TT completion routine

```

1          .SBTTL  TTCPL  -- Queue a TT completion routine
2          ;-----
3          ; Check to see if there are any input characters pending and the job
4          ; wants a completion routine executed for input characters.
5          ; If so, queue the completion request.
6          ;
7          ; Inputs:
8          ;   R1 = Job index number
9          ;
10         001222  010246  TTCPL:  MOV     R2,-(SP)
11         001224  010446          MOV     R4,-(SP)
12         ;
13         ; See if job want completion for character input
14         ;
15         001226          DISABL          ;** Disable interrupts **
16         001234  016104  00000000  MOV     LTTCLR(R1),R4  ;Does job want TT input compl routine?
17         001240  001434          BEQ     9$              ;Br if not
18         001242  016102  00000000  MOV     LINPNT(R1),R2  ;Get pointer to next char to get
19         001246  020261  00000000  CMP     R2,LINNXT(R1)  ;Are there any pending input chars?
20         001252  001427          BEQ     9$              ;Br if not
21         001254  005061  00000000  CLR     LTTCLR(R1)      ;Say we are using completion routine
22         001260          ENABL           ;** Enable interrupts **
23         ;
24         ; Get next pending input character
25         ;
26         001266          OCALL    DELCHR          ;Get next char from input buffer
27         001274  103414          BCS     8$          ;Br if no chars available
28         ;
29         ; We got a character
30         ;
31         001276  005061  00000000  CLR     LRTCHR(R1)      ;Say no read time-out pending
32         001302  042700  177400    BIC     #^C<377>,R0     ;Clear all but low-order 8 bits of char
33         001306          OCALL    CVTLC          ;Possibly convert to upper case
34         ;
35         ; Queue the completion routine
36         ;
37         001314  010064  00000000  MOV     R0,CQ$R0(R4)    ;Pass char to compl rtn in R0
38         001320  004737  00000000  CALL    QCOMPL         ;Queue the completion routine
39         001324  000402          BR      9$
40         ;
41         ; No characters available.
42         ; Leave the completion request pending.
43         ;
44         001326  010461  00000000  8$:  MOV     R4,LTTCLR(R1) ;Say compl request still pending
45         ;
46         ; Finished
47         ;
48         001332          9$:  ENABL           ;** Make sure interrupts are enabled **
49         001340  012604          MOV     (SP)+,R4
50         001342  012602          MOV     (SP)+,R2
51         001344  000207          RETURN
52         ;
53         000001          .END

```

Errors detected: 0

*** Assembler statistics

TSTTY2 -- Auxiliary terminal to MACRO V05.04 Friday 18-Dec-87 14:02 Page 8-1
TTCPL -- Queue a TT completion routine

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

\$INIT	1-31	5-47				
\$LOFCF	1-28	4-17				
\$NABRS	1-27	2-72	2-85			
\$NDIN	1-28	4-19				
\$NDINT	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*	

LTTOR	1-24	7-20	7-23*	7-45*	8-16	8-21*	8-44*
LTPAR	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					

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#					