

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
9-	1	DOCTLT -- Display a job's state information
10-	1	GTSMBF -- Get a system message buffer
11-	1	QCTMSG -- Queue ^T information to be sent to job
12-	1	BLDSTR -- Build ^T information string
13-	1	EDT routines to convert internal to ASCII format
13-	2	EDTDEC
13-	50	EDTTIM
13-	113	EDTR50 -- Decode RAD50 value to ASCII
14-	1	STATES -- Names of job states

```

1          .TITLE  TSTTY2 -- Auxiliary terminal control overlay
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5 000000    .CSECT  TSTTY2
6 000000    100100  TSTTY2: .RAD50  /TT2/          ;Overlay region id
7          ;
8          ; TSTTY2 is an auxiliary terminal control overlay.
9          ; The routines in TSTTY2 could logically go in TSTTY except
10         ; that overlay has already reached the 8Kb size limit.
11         ;
12         ; Copyright (c) 1980,1981,1982,1983,1984,1985.
13         ; S&H Computer Systems, Inc.
14         ; Nashville, Tennessee USA
15         ;
16         ; Global definitions
17         ;
18         .GLOBL  AUTSPD,DOSWIT,LINSWT,STTCPL,TTCPPL,SPPRED,DOCTLT
19         ;
20         ; Global references
21         ;
22         ;
23         ; Global definitions
24         .GLOBL  DOCTLT
25         ; Global references
26         .GLOBL  NMUMB,SNMSHD,SB$LNK,SB$TXT,SB$PNT
27         .GLOBL  LNPRIM,LMSGBF,TRNSTR,ESC,SPACE,CR
28         .GLOBL  LUNAME,SYTIMH,SYTIML,LPRG1,LPRG2,LNBLKS
29         .GLOBL  LNSBLK,LSTATE,LSW,$INCOR,LCPUHI,LCPULO,TK1VAL
30         .GLOBL  URO,QFREE,GETQ,S$TTSC,EMTMAP,CQ$PA5,LINPNT,DELCHR
31         .GLOBL  LRTCHR,CVTLC,CQ$RO,S$IOFN,$NOINT,LSW7
32         .GLOBL  $VBELL,EMTBLK,VALADW,LTTCR,EMTXIT
33         .GLOBL  CQ$JOB,CQ$RNS,LPRI,CQ$PRI,CP$STD
34         .GLOBL  CQ$CP,CQ$RTN,QCOMPL,LINNXT
35         .GLOBL  LABTIM,$RBRK,LSW10,S600,SETSPD,LMXPRM,S9600,$NABRS
36         .GLOBL  LSW9,$NOVLN,LSW2,$SUCF,$LOFCF,$NOIN,LSW3,LNPRIM
37         .GLOBL  LSECPT,FSTSL,LSTSL,INITLN,LPARNT,LSECPT,$VNOTT,LSW
38         .GLOBL  LBSPRI,VPRILO,VPRIHI,VPRIVR,LPRI,LWINDO,WINSF
39         .GLOBL  LNMAP,$INIT,WINST,LOTSPC,LOTPNT,LOTBUF,LOTEND
40         .GLOBL  TRNSTR,FORCEX,OVRHC,PR7,PSW,KPAR6,LTTPAR,INTPRI
41         .GLOBL  S19200,S4800,S3600,S2400,S1800,S1200,S300,S150,S110
42         ;
43         ; Macro definitions
44         ;
45         .MACRO  DISABL          ;DISABLE INTERRUPTS
46         BIS     #PR7,@#PSW
47         .ENDM   DISABL
48
49         .MACRO  ENABL          ;ENABLE INTERRUPTS
50         BIC     INTPRI,@#PSW
51         .ENDM   ENABL
52
53         .MACRO  OCALL  ENTADD
54         .IF     B,ENTADD
55         .ERROR  ;OCALL without entry address
56         .ENDC
57         CALL    OVRHC

```

```

58          .WORD   ENTADD
59          .ENDM   OCALL
60
61      ;
62      ; The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
63      ; terminal character buffer area. The previous contents of par6 map
64      ; register are pushed on the stack and may be restored by using the
65      ; UNMAP or UNMAPX macros.
66      ; R1 must contain the line index number of the line whose buffers
67      ; are being accessed.
68      ; The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
69      ; macros is that the X-versions are more efficient but may only be
70      ; used from within interrupt service routines where we are guaranteed
71      ; to be running on the system stack.
72      ; The TTMAP and UNMAP versions of the macros must only be
73      ; used in sections of code where the interrupts are disabled.
74      ;
75      .MACRO   TTMAPX
76      MOV      LTPAR(R1),@#KPAR6
77      .ENDM   TTMAPX
78
79      ;
80      .MACRO   UNMAPX
81      .ENDM   UNMAPX
82
83      ;
84      .MACRO   TTMAP
85      MOV      @KPAR6,MAPHLD
86      MOV      LTPAR(R1),@#KPAR6
87      .ENDM   TTMAP
88
89      ;
90      .MACRO   UNMAP
91      MOV      MAPHLD,@#KPAR6
92      .ENDM   UNMAP
93
94      ; Data areas
95      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 000000G 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 000000G 000000G  BIT     #$RBRK, LSW10(R1); Did we receive a break?
29 000036 001405          BEQ     6$         ; Br if not
30 000040 012700 000000G          MOV     #S600, R0      ; Select 600 baud
31 000044 004737 000000G          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$:  MOV     #ASTBL1, R2      ; Assume we are running at 9600 baud
37 000056 016103 000000G          MOV     LMXPRM(R1), R3 ; Get current line speed
38 000062 000303          SWAB     R3          ; Get to low-order byte
39 000064 042703 177760          BIC     #^C17, R3      ; Clear all but speed code
40 000070 020327 000000G          CMP     R3, #S9600     ; Is speed currently 9600?
41 000074 001411          BEQ     1$         ; Br if yes
42 000076 012702 000256'          MOV     #ASTBL2, R2      ; Assume speed is 600 baud
43 000102 020327 000000G          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?

```

```

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 000000G  5$:      MOV      #9600,R0      ;Select 9600 baud
66 000142 004737 000000G          CALL     SETSPD        ;Set the line speed
67                          ;
68                          ; We could not identify this character.
69                          ; Start timer which will reset the baud rate to 9600 after 10 seconds.
70                          ;
71 000146 012761 000144 000000G 8$:      MOV      #100.,LABTIM(R1);Start autobaud timer for this line
72 000154 052761 000000G 000000G      BIS      #$NABRS,LSW9(R1);Reset speed to 9600 baud after time interval
73 000162 000261          10$:      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$:      MOVB     (R2)+,R0      ;Get the correct speed code
79 000170 004737 000000G          CALL     SETSPD        ;Set the line speed
80                          ;
81                          ; Set a short timer to cause us to ignore any garbage characters
82                          ; that may be received after the autobaud character.
83                          ;
84 000174 012761 000002 000000G 11$:     MOV      #2.,LABTIM(R1) ;Ignore input chars for 0.2 seconds
85 000202 042761 000000G 000000G      BIC      #$NABRS,LSW9(R1);Clear need-reset flag
86 000210 000241          CLC              ;Signal to start the line
87                          ;
88                          ; Finished
89                          ;
90 000212 042761 000000G 000000G 9$:      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			AB1TBL == .-TSTTY2
5	000230	177	0000	ASTBL1: .BYTE 177,S19200
6	000232	176	0000	.BYTE 176,S19200
7	000234	175	0000	.BYTE 175,S19200
8	000236	172	0000	.BYTE 172,S19200
9	000240	171	0000	.BYTE 171,S19200
10	000242	162	0000	.BYTE 162,S19200
11	000244	015	0000	.BYTE 015,S9600
12	000246	146	0000	.BYTE 146,S4800
13				;; .BYTE 014,S3600
14				;; .BYTE 176,S2400
15	000250	170	0000	.BYTE 170,S2400
16				;; .BYTE 160,S1800
17	000252	000	0000	.BYTE 000,S1200
18	000254	000	000	.BYTE 0,0
19		000026		AB1LEN == .-ASTBL1
20				
21				; Table to indicate speed when we are receiving at 600 baud
22				
23	000256			AB2TBL == .-TSTTY2
24	000256	177	0000	ASTBL2: .BYTE 177,S2400
25	000260	176	0000	.BYTE 176,S1200
26	000262	172	0000	.BYTE 172,S1200
27	000264	171	0000	.BYTE 171,S1200
28	000266	170	0000	.BYTE 170,S1200
29	000270	162	0000	.BYTE 162,S1200
30				;; .BYTE 015,S600
31	000272	146	0000	.BYTE 146,S300
32				;; .BYTE 170,S150
33	000274	160	0000	.BYTE 160,S110
34	000276	140	0000	.BYTE 140,S110
35	000300	000	000	.BYTE 0,0
36		000024		AB2LEN == .-ASTBL2

```

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    000000G 000000G      BIT     #$NOVLN,LSW2(R1);Can job use virtual lines?
16 000316    001060      BNE     20$          ;Br if not
17 000320    032761    000000C 000000G      BIT     <#$SUCF!$LOFCF>,LSW9(R1);Doing logon/logoff processing?
18 000326    001054      BNE     20$          ;Br if yes
19 000330    032761    000000G 000000G      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    000000G      MOV     LNPRIM(R1),R2 ;Get primary line #
31 000350    016203    000000G      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    000000G      MOV     #FSTSL,R3      ;Get # of first virtual line
41 000366    020327    000000G 3$: CMP     R3,#LSTSL      ;Checked all?
42 000372    101032      BHI     20$          ;Br if yes
43 000374    005763    000000G      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

```

DOSWIT -- Switch to a virtual 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              MOV     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
```

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    000000G 000000G    BIS     #$VNOTT,LSW(R1) ; Say not connected to terminal
20         ;
21         ; Reduce execution priority of job we are disconnecting from
22         ;
23 000510    004737    000776'    CALL     SPPRED          ; Reduce execution priority of discon proc
24         ;
25         ; Clear flag that says bell has been rung for virtual line wait
26         ; condition for line we are switching away from.
27         ;
28 000514    042761    000000G 000000G    BIC     #$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    000000G          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    000000G    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    000000G          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    000000G    1$: MOV     R1,LNMAP(R2)      ; SWITCH LINE CONTROL TO VIRTUAL LINE
47 000564    032761    000000G 000000G    BIT     #$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    000000G          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$:      DISABL      ;** DISABLE **
64 000620      012704  000006      MOV      #6,R4      ;# OF CHARS WE WANT TO INSERT
65 000624      026104  000000G     CMP      LOTSPC(R1),R4 ;ROOM FOR OUT CHARS?
66 000630      002434          BLT      3$      ;IF NOT THEN SKIP INSERT
67 000632      160461  000000G     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'     MOV      R5,<NLMSG+3> ;MOVE INTO MESSAGE
71 000652      016102  000000G     MOV      LOTPNT(R1),R2 ;POINT TO HEAD OF OUTPUT LIST
72 000656          TTMAP          ;MAP TO TT BUFFER AREA
73 000672      020261  000000G     5$:      CMP      R2,LOTBUF(R1) ;AT HEAD OF BUFFER?
74 000676      101002          BHI      4$      ;BRANCH IF NOT
75 000700      016102  000000G     MOV      LOTEND(R1),R2 ;WRAP AROUND TO END
76 000704      112342          4$:      MOV      (R3)+,-(R2) ;MOVE MESSAGE TO USER'S BUFFER
77 000706      077407          SOB      R4,5$      ;LOOP IF MORE TO DO
78 000710      010261  000000G     MOV      R2,LOTPNT(R1) ;UPDATE CHARACTER POINTER
79 000714          UNMAP          ;RESTORE MAPPING
80 000722      3$:      ENABL      ;** ENABLE **
81      ;
82      ; Set up info for job we are switching to
83      ;
84 000730      042761  000000G 000000G 9$:      BIC      #$VNOTT,LSW(R1) ;SAY RECONNECTED TO TERMINAL
85 000736      116161  000000G 000000G     MOV      LBSPRI(R1),LPRI(R1);SET CURRENT PRI = BASE PRI
86 000744      004777  000000G     CALL      @TRNSTR      ;TRY TO START PRINTING
87 000750      004737  000000G     CALL      FORCEX      ;FORCE EXECUTION OF NEW LINE
88      ;
89      ; Finished
90      ;
91 000754      012605          6$:      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

```

SPPRED -- Reduce execution priority for disconnect process

```

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     LBSPRI(R1), R2      ; GET BASE PRIORITY FOR JOB
11         001004 120237 0000000 CMPB    R2, VPRILO      ; IS THIS A FIXED PRIORITY JOB?
12         001010 101414      BLOS     7$                ; BR IF YES
13         001012 120237 0000000 CMPB    R2, VPRIHI      ; REAL-TIME JOB?
14         001016 103011      BHIS     7$                ; BR IF YES
15         001020 113700 0000000 MOV     VPRIVR, R0        ; GET AMT TO REDUCE FOR DISCONNECTED JOBS
16         001024 160002      SUB      R0, R2             ; REDUCE FOR DISCONNECTED JOB
17         001026 120237 0000000 CMPB    R2, VPRILO      ; DID PRIORITY GO OUT OF NORMAL RANGE?
18         001032 003003      BGT      7$                ; BR IF NOT
19         001034 113702 0000000 MOV     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

```

STTCPL -- Set completion routine for TT activation

```

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   000000G   CLR     URO           ; Assume no completion routine now
20 001060   016201   000000G   MOV     LTTCR(R2),R1 ; Get addr of completion Q element
21 001064   001407       BEQ     1$             ; Br if none
22 001066   016137   000000G 000000G   MOV     CQ$RTN(R1),URO ; Return old compl rtn address in R0
23 001074   005062   000000G   CLR     LTTCR(R2)   ; Say no completion Q element now
24 001100   004737   000000G   CALL    QFREE       ; Release the queue element
25         ;
26         ; See if the address of a new completion routine was specified
27         ;
28 001104   013700   000002G 1$:   MOV     EMTBLK+2,R0 ; Did he specify a compl rtn address?
29 001110   001442       BEQ     9$             ; Br if not
30 001112   004737   000000G   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   000000G   CALL    GETQ        ; Get a queue element (address in R1)
35 001122   010104       MOV     R1,R4           ; Get address of queue element
36 001124   110264   000000G   MOVB    R2,CQ$JOB(R4) ; Set job number in Q element
37 001130   112764   000000G 000000G   MOVB    #S$TTSC,CQ$RNS(R4); Set execution state for job
38 001136   032762   000000G 000000G   BIT     #$NOINT,LSW7(R2); Is job running non-interactively?
39 001144   001403       BEQ     2$             ; Br if not
40 001146   112764   000000G 000000G   MOVB    #S$IDFN,CQ$RNS(R4); Set non-interactive execution state
41 001154   116264   000000G 000000G 2$:   MOVB    LPRI(R2),CQ$PRI(R4); Set execution priority
42 001162   112764   000000G 000000G   MOVB    #CP$STD,CQ$CP(R4); Set compl routine class priority
43 001170   013764   000002G 000000G   MOV     EMTBLK+2,CQ$RTN(R4); Set address of compl routine
44 001176   013764   000000G 000000G   MOV     EMTMAP,CQ$PA5(R4); Set EMT entry mapping for par 5
45 001204   010462   000000G   MOV     R4,LTTCR(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   001222'   CALL    TTCPL      ; See if we need to do compl rtn now
52         ;
53         ; Finished
54         ;
55 001216   000137   000000G 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
11         001224  010446
12         ;
13         ; See if job want completion for character input
14         ;
15         001226          DISABL          ;** Disable interrupts **
16         001234  016104  000000G      MOV    LTTCR(R1),R4      ;Does job want TT input compl routine?
17         001240  001434          BEQ     9$                    ;Br if not
18         001242  016102  000000G      MOV    LINPNT(R1),R2      ;Get pointer to next char to get
19         001246  020261  000000G      CMP    R2,LINNXT(R1)      ;Are there any pending input chars?
20         001252  001427          BEQ     9$                    ;Br if not
21         001254  005061  000000G      CLR    LTTCR(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  000000G      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  000000G      MOV    R0,CQ$R0(R4)        ;Pass char to compl rtn in R0
38         001320  004737  000000G      CALL   QCOMPL              ;Queue the completion routine
39         001324  000402          BR     9$
40         ;
41         ; No characters available.
42         ; Leave the completion request pending.
43         ;
44         001326  010461  000000G      8$:  MOV    R4,LTTCR(R1)    ;Say compl request still pending
45         ;
46         ; Finished
47         ;
48         001332          9$:  ENABL          ;** Make sure interrupts are enabled **
49         001340  012604          MOV    (SP)+,R4
50         001342  012602          MOV    (SP)+,R2
51         001344  000207          RETURN

```

DOCTLT -- Display a job's state information

```

1          .SBTTL  DOCTLT -- Display a job's state information
2          ;-----
3          ; This routine is invoked when a job types ^T. The job does not
4          ; have to be resident. The format of the display is:
5          ;
6          ;      <CR>          ;in case the cursor is not at the left of the screen
7          ;      job number    ;virtual job number (may be secondary line)
8          ;      job name      ;LUNAME
9          ;      wall time     ;SYTIMH,SYTIML
10         ;      program name   ;LPRG1,LPRG2
11         ;      job size       ;LNBLKS+LNSBLK in Kb
12         ;      state          ;LSTATE
13         ;      CPU time       ;LCPUHI,LCPULO
14         ;
15         ; Inputs:
16         ;      R1 contains job index
17         ;
18 DOCTLT:
19         MOV     R2,-(SP)
20         CALL    QTSMBF ; Try to get a system message buffer (ptr in R2)
21         BCS     9$     ; Just ignore ^T if no message buffer available
22         MOV     R2,-(SP) ; Save message ptr
23         MOV     SB$PNT(R2),R2 ; Point to text buffer
24         CALL    BLDSTR ; Build the output string directly into message buffer
25         MOV     (SP)+,R2 ; Restore message ptr
26         CALL    QCTMSG ; Queue the ^T message for display
27         MOV     (SP)+,R2
28         RETURN

```

GTSMBF -- Get a system message buffer

```

1          .SBTTL  GTSMBF -- Get a system message buffer
2          ;-----
3          ; Get a free system message buffer block. If none are available,
4          ; return with carry set.
5          ;
6          ; Outputs:
7          ; R2      ptr to system message block
8          ;
9 001402    GTSMBF: DISABL      ;;; *** DISABLE ***
10 001410    TST      NMUMB      ;;; Are there any available?
11 001414    BEQ      B$         ;;; If not, return with error
12 001416    MOV      SNMSHD,R2   ;;; Point to first free buffer block
13 001422    BEQ      B$         ;;; If end of list, return with error
14          ;
15          ; We have a free system message buffer. Unlink from chain.
16          ;
17 001424    DEC      NMUMB      ;;; One less available
18 001430    MOV      SB$LNK(R2),SNMSHD ;;; Remove from free list
19 001436    ENABL      ;;; *** ENABLE ***
20          ;
21 001444    MOV      R2,-(SP)    ; Copy buffer block ptr
22 001446    ADD      #SB$TXT,(SP) ; Point to text area
23 001452    MOV      (SP)+,SB$PNT(R2) ; Init text pointer
24 001456    CLC              ; No errors
25 001460    BR       9$         ; return
26          ;
27          ; No free message buffers.
28          ;
29 001462    B$:      ENABL      ;;; *** ENABLE ***
30 001470    SEC              ; Signal error
31          ;
32 001472    9$:      RETURN

```

QCTMSG -- Queue ^T information to be sent to job

```

1          .SBTTL  QCTMSG -- Queue ^T information to be sent to job
2          ;-----
3          ; Inputs:
4          ;       R1 index of job to receive message (same as sender)
5          ;       R2 address of system message buffer
6          ;
7 001474 010046 QCTMSG: MOV      R0,-(SP)
8 001476 010146      MOV      R1,-(SP)          ; Save job index, need to work on primary
9          ;
10         DISABL          ;;; *** DISABLE ***
11 001506 016101 000000G    MOV      LNPRIM(R1),R1      ;;; Get physical line index
12 001512 016100 000000G    MOV      LMSGBF(R1),R0      ;;; Get ptr to head of job's message list
13 001516 001411          BEQ      3$                   ;;; Br if no pending messages for job
14 001520 005760 000000G    1$:    TST      SB$LNK(R0)    ;;; Is this the end of the list?
15 001524 001403          BEQ      2$                   ;;; Br if yes
16 001526 016000 000000G    MOV      SB$LNK(R0),R0      ;;; If not, point to next message
17 001532 000772          BR        1$                   ;;; And keep searching for end
18 001534 010260 000000G    2$:    MOV      R2,SB$LNK(R0) ;;; Add current message to end of list
19 001540 000402          BR        4$                   ;;;
20 001542 010261 000000G    3$:    MOV      R2,LMSGBF(R1) ;;; Say current message is first
21 001546 005062 000000G    4$:    CLR      SB$LNK(R2)   ;;; Say we are at end of list
22 001552          ENABL          ;;; *** ENABLE ***
23          ;
24          ; Try to start transmission to the line
25          ;
26 001560 012601          MOV      (SP)+,R1              ; Restore index of job which hit ^T
27 001562 004777 000000G    CALL     @TRNSTR            ; Initiate transmission
28          ;
29 001566 012600          MOV      (SP)+,R0
30 001570 000207          RETURN

```

```

1          .SBTTL  BLDSTR -- Build ^T information string
2          -----
3          ; Create ^T information in ASCII format into message buffer using
4          ; system table information.
5          ;
6          ; Inputs:
7          ;     R1 job index
8          ;     R2 ptr to beginning of text buffer
9          ;
10         BLDSTR: MOV     R0, -(SP)
11         MOV     R3, -(SP)
12         MOV     R4, -(SP)
13         MOV     R5, -(SP)
14
15         MOV     R0, 00000000
16         ;
17         ; Job number
18         ;
19         MOV     R1, R0
20         ASR     R0
21         CLR     R3
22         CALL    EDTDEC
23         MOV     R0, #SPACE, (R2)+
24         MOV     R0, #SPACE, (R2)+
25         ;
26         ; Job name
27         ;
28         MOV     R2, -(SP)
29         MOV     R1, R3
30         MUL     #6, R3
31         ADD     #LUNAME, R3
32         MOV     #12, R4
33         1$: MOV     (R3)+, R0
34         BLE     14$
35         CMP     R0, #ESC
36         BLO     12$
37         MOV     R0, (R2)+
38         12$: SOB     R4, 1$
39         ; Now remove trailing spaces
40         14$: CMP     R2, (SP)
41         BLOS    15$
42         CMP     -(R2), #SPACE
43         BEQ     14$
44         INC     R2
45         15$: MOV     #SPACE, (R2)+
46         MOV     #SPACE, (R2)+
47         TST     (SP)+
48         ; Discard saved string ptr
49         ;
50         ; Wall clock time
51         ;
52         MOV     SYTIMH, R4
53         MOV     SYTIML, R5
54         CALL    EDTTIM
55         MOV     #SPACE, -2(R2)
56         MOV     #SPACE, -1(R2)
57         ; Program name

```

BLDSTR -- Build ^T information string

```

58
59 001744 016100 000000G      MOV      LPRG1(R1),R0      ;Get program name 1-3
60 001750 004737 002450'      CALL      EDTR50          ;Stuff into output
61 001754 016100 000000G      MOV      LPRG2(R1),R0      ;Get program name 4-6
62 001760 001402              BEQ        2$              ;Br if less than 4 char name
63 001762 004737 002450'      CALL      EDTR50
64 001766 126227 177776 000000G 2$:  CMPB      -2(R2),#SPACE      ;Two spaces already?
65 001774 001410              BEQ        22$             ;Don't add any if so
66 001776 126227 177777 000000G      CMPB      -1(R2),#SPACE      ;Last char space?
67 002004 001402              BEQ        21$             ;Only need 1 if so
68 002006 112722 000000G      MOVB      #SPACE,(R2)+
69 002012 112722 000000G      21$:  MOVB      #SPACE,(R2)+
70 002016              22$:
71
72      ; Job size
73
74 002016 016100 000000G      MOV      LNBLKS(R1),R0      ;Get static job size (in pages)
75 002022 066100 000000G      ADD      LNSBLK(R1),R0      ;And add PLAS region size
76 002026 005200              INC      R0                ;Round up to nearest KB
77 002030 006200              ASR      R0                ;Convert pages to Kb
78 002032 005003              CLR      R3                ;No filler chars
79 002034 004737 002210'      CALL      EDTDEC          ;Stuff size in Kb into output
80 002040 112722 000113      MOVB      #'K,(R2)+          ;Append "Kb"
81 002044 112722 000142      MOVB      #'b,(R2)+
82 002050 112722 000000G      MOVB      #SPACE,(R2)+
83 002054 112722 000000G      MOVB      #SPACE,(R2)+
84
85      ; Job state
86
87 002060 016100 000000G      MOV      LSTATE(R1),R0      ;Get job state
88 002064 020027 000036      CMP      R0,#MXSTAT        ;Last state we know about
89 002070 101401              BLOS      3$              ;Br if in known state space
90 002072 005000              CLR      R0                ;Else make it unknown
91 002074 006300              3$:  ASL      R0                ;Convert to state index
92 002076 016000 002566'      MOV      SNTABL(R0),R0      ;Get ptr to state name string
93 002102 112022              32$:  MOVB      (R0)+,(R2)+      ;Copy state name string to output
94 002104 003376              BGT      32$              ;0 or 200 terminates
95 002106 032761 000000G 000000G      BIT      #$INCOR,LSW(R1) ;In memory now?
96 002114 001005              BNE      34$              ;Br if so
97 002116 012703 003326'      MOV      #SWPTXT,R3        ;Job is swapped out, point to swap text
98 002122 005302              DEC      R2                ;Point back to trailing NUL
99 002124 112322              33$:  MOVB      (R3)+,(R2)+      ; and append
100 002126 003376              BGT      33$
101 002130 112762 000000G 177777 34$:  MOVB      #SPACE,-1(R2) ;Copy space over trailing nul
102 002136 112722 000000G      MOVB      #SPACE,(R2)+
103
104      ; CPU time
105
106 002142 112722 000103      MOVB      #'C,(R2)+          ;Put "CPU=" into output buffer
107 002146 112722 000120      MOVB      #'P,(R2)+
108 002152 112722 000125      MOVB      #'U,(R2)+
109 002156 112722 000075      MOVB      #'=(R2)+
110 002162 016104 000000G      MOV      LCPUHI(R1),R4      ;Get job cpu time in ticks
111 002166 016105 000000G      MOV      LCPULO(R1),R5
112 002172 004737 002310'      CALL      EDTTIM          ;Move time into output buffer
113
114      ; Done. NOTE: cursor is left at end of ^T string.

```

```
115  
116 002176 012605      MOV      (SP)+, R5  
117 002200 012604      MOV      (SP)+, R4  
118 002202 012603      MOV      (SP)+, R3  
119 002204 012600      MOV      (SP)+, R0  
120 002206 000207      RETURN
```

```

1          .SBTTL  EDT routines to convert internal to ASCII format
2          .SBTTL  EDTDEC
3          -----
4          ;
5          ; Inputs:
6          ;   R0 number to be converted
7          ;   R3 low byte min field width, high byte filler char
8          ;
9          ; Outputs:
10         ;   number string is encoded at location pointed to by R2
11         ;   R2 left at byte immediately following last digit
12         ;   R0 is destroyed
13         ;
14 002210 010346 EDTDEC: MOV     R3, -(SP)
15 002212 010446      MOV     R4, -(SP)
16 002214 010546      MOV     R5, -(SP)
17         ;
18 002216 010005      MOV     R0, R5      ; Copy input value into R5
19 002220 005000      CLR      R0        ; Counter for # of digits used
20         ; Remainder is in R5
21 002222 005004 1$:   CLR      R4        ; Set for divide
22 002224 071427 000012 DIV     #10, R4   ; Split out low digit R5, quotient in R4
23 002230 110546      MOVB    R5, -(SP)   ; Save binary digit on stack
24 002232 005200      INC     R0        ; Count a digit saved
25 002234 010405      MOV     R4, R5     ; Copy quotient for next divide
26 002236 001371      BNE     1$        ; Loop until insignificant
27         ; Digits are on stack
28 002240 110305      MOVB    R3, R5     ; Get minimum field width
29 002242 160005      SUB     R0, R5     ; Did we fill minimum field width?
30 002244 003407      BLE     3$        ; Br if so
31         ; Didn't fill field, pad with filler chars
32 002246 000303      SWAB    R3        ; Get filler char to low byte
33 002250 105703      TSTB    R3        ; Is there a filler char?
34 002252 001002      BNE     2$        ; Yes, use it
35 002254 012703 000040 MOV     #40, R3   ; If not, fill with spaces
36 002260 110322 2$:   MOVB    R3, (R2)+ ; Pad with leading filler chars
37 002262 077502      SOB     R5, 2$    ; As many as we need
38         ; Field is padded, now retrieve digits off stack and drop leading 0's
39 002264 112605 3$:   MOVB    (SP)+, R5 ; Get next digit
40 002266 062705 000060 ADD     #'0, R5   ; Convert to ascii
41 002272 110522      MOVB    R5, (R2)+ ; Stuff into output
42 002274 077005      SOB     R0, 3$    ; Thru all digits
43 002276 105012      CLRB    (R2)     ; Always terminate string
44         ; Done
45 002300 012605      MOV     (SP)+, R5
46 002302 012604      MOV     (SP)+, R4
47 002304 012603      MOV     (SP)+, R3
48 002306 000207      RETURN
49         ;
50         .SBTTL  EDTTIM
51         -----
52         ; Convert a 32-bit tick count into HH:MM:SS.T (.1 sec precision)
53         ;
54         ; Inputs:
55         ;   R2 ptr to string output buffer
56         ;   R4 time high
57         ;   R5 time low

```

```

58
59      ; Outputs:
60      ;   hh:mm:ss.t encoded at location pointed to by R2
61      ;   R2 is left pointing at byte immediately following tenths
62
63 002310 010046
64 002312 010346
65 002314 010446
66 002316 010546
67
68      ;
69      ; Start by splitting ticks into two 1-word parts by dividing by
70      ; number of ticks per minute. This leaves two parts which will each
71      ; fit into 1 word (24.*60.=1440. and 60.*60.=3600.)
72      ;
73      ;   MOV      TK1VAL,R3      ;Get back # ticks/.1 sec
74      ;   MUL      #600.,R3      ;Convert to # .1s/minute
75      ;   DIV      R3,R4         ;Split time into hrs*min and sec*ticks
76
77      ; Now convert to hours and minutes
78      ;
79      ;   MOV      R5,-(SP)       ;Save sec*ticks
80      ;   MOV      R4,R5         ;Move hrs*min for divide
81      ;   CLR      R4            ;Set for divide
82      ;   DIV      #60.,R4       ;Split hours and minutes
83      ;   MOV      R4,R0         ;Get hours
84      ;   CLR      R3            ;Free format
85      ;   CALL     EDTDEC        ;Display hours
86      ;   MOVB     #'',(R2)+
87      ;   MOV      R5,R0         ;Get minutes
88      ;   MOV      #400*0+2,R3   ;Format 2 digits, leading 0
89      ;   CALL     EDTDEC        ;Display minutes
90      ;   MOVB     #'',(R2)+
91      ;   MOV      (SP)+,R5       ;Retrieve seconds and 10ths
92
93      ; Now display seconds and tenths
94      ;
95      ;   CLR      R4            ;Set for divide
96      ;   DIV      #60.,R4       ;Split seconds and ticks
97      ;   MOV      R4,R0         ;Get seconds
98      ;   CALL     EDTDEC        ;Display seconds (format still in R3)
99      ;   MOVB     #'',(R2)+
100     ;   CLR      R4            ;Set for divide
101     ;   DIV      TK1VAL,R4      ;Split into tenths of seconds
102     ;   ADD      #'0,R4        ;Has to be 0-9
103     ;   MOVB     R4,(R2)+
104     ;   CLRB     (R2)          ;Always terminate
105
106     ; Done!
107     ;
108     ;   MOV      (SP)+,R5
109     ;   MOV      (SP)+,R4
110     ;   MOV      (SP)+,R3
111     ;   MOV      (SP)+,R0
112     ;   RETURN
113
114     ; -----
115     ;   SBTTL      EDTR50 -- Decode RAD50 value to ASCII

```

```

115      ; Inputs:
116      ;      R0 value to be decoded
117      ;      R2 ptr to output location
118      ;
119      EDTR50:
120      MOV     R4,-(SP)
121      MOV     R5,-(SP)
122      MOV     R0,R5      ; Copy value to be decoded
123      CLR     R4          ; Set for divide
124      DIV     #50*50,R4   ; Split out high letter
125      MOVB    R5OTBL(R4),(R2)+ ; Decode A. to output
126      CLR     R4          ; Set for next divide
127      DIV     #50,R4      ; Split middle and last letters
128      MOVB    R5OTBL(R4),(R2)+ ; Decode .B. to output
129      MOVB    R5OTBL(R5),(R2)+ ; Decode ..C to output
130      CLRB    (R2)        ; Always terminate string
131      MOV     (SP)+,R5
132      MOV     (SP)+,R4
133      RETURN
134      .NLIST BEX
135      002516      040      101      102 R5OTBL: .ASCII / ABCDEFGHIJKLMNOPQRSTUVWXYZ$. ?0123456789/
136      .LIST BEX
137      .EVEN

```

STATES -- Names of job states

```

1
2
3 002566 003316'
4 002570 002664'
5 002572 002664'
6 002574 002677'
7 002576 002714'
8 002600 002732'
9 002602 002752'
10 002604 002770'
11 002606 003003'
12 002610 003021'
13 002612 003021'
14 002614 003037'
15 002616 003047'
16 002620 003047'
17 002622 003066'
18 002624 003077'
19 002626 003110'
20 002630 003121'
21 002632 003132'
22 002634 003143'
23 002636 003153'
24 002640 003164'
25 002642 003175'
26 002644 003206'
27 002646 003217'
28 002650 003227'
29 002652 003240'
30 002654 003251'
31 002656 003262'
32 002660 003274'
33 002662 003305'
34 000036
35
36
37 002664 122 145 141 1$:
38 002676 000 137 2$:
39 002677 124 124 137 3$:
40 002714 124 124 137 4$:
41 002732 124 124 137 5$:
42 002752 111 156 164 6$:
43 002770 124 151 155 7$:
44 003003 124 124 137 10$:
45 003021 111 057 117 11$:
46 003036 000 12$:
47 003037 103 157 155 13$:
48 003047 114 157 167 14$:
49 003065 000 15$:
50 003066 127 141 151 16$:
51 003077 127 141 151 17$:
52 003110 127 141 151 20$:
53 003121 127 141 151 21$:
54 003132 127 141 151 22$:
55 003143 127 141 151 23$:
56 003153 127 141 151 24$:
57 003164 127 141 151 25$:

.SBTTL STATES -- Names of job states
-----
SNTABL: .WORD 50$ ;Unknown state!!!
        .WORD 1$ ;S$RT
        .WORD 1$ ;S$$RT
        .WORD 3$ ;S$TTSC
        .WORD 4$ ;S$TTFN
        .WORD 5$ ;S$OTFN
        .WORD 6$ ;S$HICP
        .WORD 7$ ;S$TWFN
        .WORD 10$ ;S$OTLO
        .WORD 11$ ;S$IOFN
        .WORD 11$ ;S$$HIP
        .WORD 13$ ;S$CPU
        .WORD 14$ ;S$LOW
        .WORD 14$ ;S$$RUN
        .WORD 16$ ;S$NEDQ
        .WORD 17$ ;S$QMIO
        .WORD 20$ ;S$QCCB
        .WORD 21$ ;S$QCXB
        .WORD 22$ ;S$QUSR
        .WORD 23$ ;S$IOWT
        .WORD 24$ ;S$OTWT
        .WORD 25$ ;S$SFWT
        .WORD 26$ ;S$WSMB
        .WORD 27$ ;S$SPDB
        .WORD 30$ ;S$INWT
        .WORD 31$ ;S$QSPD
        .WORD 32$ ;S$SPCB
        .WORD 33$ ;S$MSWT
        .WORD 34$ ;S$SPND
        .WORD 35$ ;S$TMWT
        .WORD 36$ ;S$WFM
MXSTAT = <.-SNTABL-2>/2 ;Define last state number we understand
;
.NLIST BEX
        .ASCIZ /Real-time/
        .ASCIZ // ;S$$RT
        .ASCIZ /TT_sngl_done/
        .ASCIZ /TT_input_done/
        .ASCIZ /TT_output_empty/
        .ASCIZ /Interact_comp/
        .ASCIZ /Timer_done/
        .ASCIZ /TT_output_low/
        .ASCIZ "I/O_complete"
        .ASCIZ // ;S$$HIP
        .ASCIZ /Compute/
        .ASCIZ /Low_prio_comp/
        .ASCIZ // ;S$$RUN
        .ASCIZ /Wait-IDQ/
        .ASCIZ /Wait-MIO/
        .ASCIZ /Wait-CSH/
        .ASCIZ /Wait-CXT/
        .ASCIZ /Wait-USR/
        .ASCIZ /Wait-ID/
        .ASCIZ /Wait-TTO/
        .ASCIZ /Wait-SHF/

```

STATES -- Names of job states

58	003175	127	141	151	26\$:	.ASCIZ	/Wait-SMB/
59	003206	127	141	151	27\$:	.ASCIZ	/Wait-SPF/
60	003217	127	141	151	30\$:	.ASCIZ	/Wait-TI/
61	003227	127	141	151	31\$:	.ASCIZ	/Wait-SDD/
62	003240	127	141	151	32\$:	.ASCIZ	/Wait-SPC/
63	003251	127	141	151	33\$:	.ASCIZ	/Wait-MSG/
64	003262	123	165	163	34\$:	.ASCIZ	/Suspended/
65	003274	127	141	151	35\$:	.ASCIZ	/Wait-TMR/
66	003305	127	141	151	36\$:	.ASCIZ	/Wait-MEM/
67	003316	077	123	164	50\$:	.ASCIZ	/?State?/
68	003326	055	123	167	SWPTXT:	.ASCIZ	/-Swap/
69						.LIST	BEX
70						.EVEN	
71							
72	000001					.END	

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 8342 Words (33 Pages)
 Size of core pool: 18176 Words (71 Pages)
 Operating system: RT-11

Elapsed time: 00:00:20.28
 ,LP:TSTTY2=DK:TSTTY2/C/N:SYM

\$INCOR	1-29	12-95					
\$INIT	1-39	5-47					
\$LOFCF	1-36	4-17					
\$NABRS	1-35	2-72	2-85				
\$NOIN	1-36	4-19					
\$NOINT	1-31	7-38					
\$NOVLN	1-36	4-15					
\$RBRK	1-35	2-28	2-90				
\$SUCF	1-36	4-17					
\$VBELL	1-32	5-28					
\$VNOTT	1-37	5-19	5-84				
AB1LEN	3-19#						
AB1TBL	3-4#						
AB2LEN	3-36#						
AB2TBL	3-23#						
ASTBL1	2-36	3-5#	3-19				
ASTBL2	2-42	3-24#	3-36				
AUTSPD	1-18	2-14#					
BLDSTR	9-24	12-10#					
CP\$STD	1-33	7-42					
CQ\$CP	1-34	7-42*					
CQ\$JOB	1-33	7-36*					
CQ\$PA5	1-30	7-44*					
CQ\$PRI	1-33	7-41*					
CQ\$RO	1-31	8-37*					
CQ\$RNS	1-33	7-37*	7-40*				
CQ\$RTN	1-34	7-22	7-43*				
CR	1-27	12-15					
CVTLC	1-31	8-33					
DELCHR	1-30	8-26					
DOCTLT	1-18	1-24	9-18#				
DOSWIT	1-18	4-9#					
EDTDEC	12-22	12-79	13-14#	13-84	13-88	13-97	
EDTR50	12-60	12-63	13-119#				
EDTTIM	12-53	12-112	13-63#				
EMTBLK	1-32	7-28	7-43				
EMTMAP	1-30	7-44					
EMTXIT	1-32	7-55					
ESC	1-27	12-35					
FORCEX	1-40	5-87					
FSTSL	1-37	4-40					
GETQ	1-30	7-34					
GTSMBF	9-20	10-9#					
INITLN	1-37	4-55					
INTPRI	1-40	5-80	8-22	8-48	10-19	10-29	11-22
KPAR6	1-40	5-72	5-72*	5-79*			
LABTIM	1-35	2-22	2-71*	2-84*			
LBSPRI	1-38	5-85	6-10				
LCPUHI	1-29	12-110					
LCPULO	1-29	12-111					
LINNXT	1-34	8-19					
LINPNT	1-30	8-18					
LINSWT	1-18	4-67	5-11#				
LMSGBF	1-27	11-12	11-20*				
LMXPRM	1-35	2-37					
LNBLKS	1-28	12-74					

LNMAP	1-39	5-46*																
LNPRIM	1-27	1-36	4-30	4-43	4-58	4-63*	5-39	11-11										
LNSBLK	1-29	12-75																
LOTBUF	1-39	5-73																
LOTEND	1-39	5-75																
LOTPNT	1-39	5-71	5-78*															
LOTSPC	1-39	5-65	5-67*															
LPARNT	1-37	4-59*																
LPRG1	1-28	12-59																
LPRG2	1-28	12-61																
LPRI	1-33	1-38	5-85*	6-21*	7-41													
LRTCHR	1-31	8-31*																
LSECPT	1-37	1-37	4-31	4-60	5-43													
LSTATE	1-29	12-87																
LSTSL	1-37	4-41																
LSW	1-29	1-37	5-19*	5-47	5-84*	12-95												
LSW10	1-35	2-28	2-90*															
LSW2	1-36	4-15																
LSW3	1-36	4-19																
LSW7	1-31	7-38																
LSW9	1-36	2-72*	2-85*	4-17	5-28*													
LTTCR	1-32	7-20	7-23*	7-45*	8-16	8-21*	8-44*											
LTTPAR	1-40	5-72																
LUNAME	1-28	12-31																
LWINDO	1-38	5-33	5-53															
MAPHLD	1-94#	5-72*	5-79															
MXSTAT	12-88	14-34#																
NLMSG	5-68	5-70*	5-98#															
NMUMB	1-26	10-10	10-17*															
OVRHC	1-40	4-55	5-35	5-55	8-26	8-33												
PR7	1-40	5-63	8-15	10-9	11-10													
PSW	1-40	5-63*	5-80*	8-15*	8-22*	8-48*	10-9*	10-19*	10-29*	11-10*	11-22*							
QCOMPL	1-34	8-38																
QCTMSG	9-26	11-7#																
QFREE	1-30	7-24																
R50TBL	13-125	13-128	13-129	13-135#														
S#IOFN	1-31	7-40																
S#TTSC	1-30	7-37																
S110	1-41	3-33	3-34															
S1200	1-41	3-17	3-25	3-26	3-27	3-28	3-29											
S150	1-41																	
S1800	1-41																	
S19200	1-41	3-5	3-6	3-7	3-8	3-9	3-10											
S2400	1-41	3-15	3-24															
S300	1-41	3-31																
S3600	1-41																	
S4800	1-41	3-12																
S600	1-35	2-30	2-43															
S9600	1-35	2-40	2-65	3-11														
SB\$LNK	1-26	10-18	11-14	11-16	11-18*	11-21*												
SB\$PNT	1-26	9-23	10-23*															
SB\$TXT	1-26	10-22																
SETSPD	1-35	2-31	2-66	2-79														
SNMSHD	1-26	10-12	10-18*															
SNTABL	12-92	14-3#	14-34															
SPACE	1-27	12-23	12-24	12-42	12-45	12-46	12-54	12-55	12-64	12-66	12-68	12-69						

	12-82	12-83	12-101	12-102
SPPRED	1-18	5-23	6-9#	
STTCPL	1-18	7-14#		
SWPTXT	12-97	14-68#		
SYTIMH	1-28	12-51		
SYTIML	1-28	12-52		
TK1VAL	1-29	13-72	13-100	
TRNSTR	1-27	1-40	5-86	11-27
TSTTY2	1-6#	3-4	3-23	
TTCPL	1-18	7-51	8-10#	
URO	1-30	7-19*	7-22*	
VALADW	1-32	7-30		
VPRIHI	1-38	6-13		
VPRILO	1-38	6-11	6-17	6-19
VPRIVR	1-38	6-15		
WINSF	1-38	5-35		
WINST	1-39	5-55		

TSTTY2 -- Auxiliary terminal co MACRO V05.05 Wednesday 18-Jan-89 10:36 Page M-1
Cross reference table (CREF V05.05)

DISABL	1-45#	5-63	8-15	10-9	11-10		
ENABL	1-49#	5-80	8-22	8-48	10-19	10-29	11-22
OCALL	1-53#	4-55	5-35	5-55	8-26	8-33	
TTMAP	1-83#	5-72					
TTMAPX	1-74#						
UNMAP	1-88#	5-79					
UNMAPX	1-80#						