Table of contents

```
                                        .TITLE  TSTTY2 -- Auxiliary terminal control overlay
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
000000                                  .CSECT  TSTTY2
000000  100100                  TSTTY2: .RAD50  /TT2/           ;Overlay region id
                                ;
                                ;  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, DOCTLT
                                ;
                                ;  Global references
                                ;
                                ;
                                ; Global definitions
                                        .GLOBL  DOCTLT
                                ; Global references
                                        .GLOBL  NMUMB, SNMSHD, SB$LNK, SB$TXT, SB$PNT
                                        .GLOBL  LNPRIM, LMSGBF, TRNSTR, ESC, SPACE, CR
                                        .GLOBL  LUNAME, SYTIMH, SYTIML, LPRG1, LPRG2, LNBLKS
                                        .GLOBL  LNSBLK, LSTATE, LSW, $INCOR, LCPUHI, LCPULO, TK1VAL
                                        .GLOBL  URO, QFREE, GETQ, S$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                          AB1TBL == .-TSTTY2
000230   177     000G                   ASTBL1: .BYTE   177,S19200
000232   176     000G                           .BYTE   176,S19200
000234   175     000G                           .BYTE   175,S19200
000236   172     000G                           .BYTE   172,S19200
000240   171     000G                           .BYTE   171,S19200
000242   162     000G                           .BYTE   162,S19200
000244   015     000G                           .BYTE   015,S9600
000246   146     000G                           .BYTE   146,S4800
                                        ;;             .BYTE   014,S3600
                                        ;;             .BYTE   176,S2400
000250   170     000G                           .BYTE   170,S2400
                                        ;;             .BYTE   160,S1800
000252   000     000G                           .BYTE   000,S1200
000254   000     000                            .BYTE   0,0
        000026                          AB1LEN == .-ASTBL1
                                        ;
                                        ;  Table to indicate speed when we are receiving at 600 baud
                                        ;
        000256                          AB2TBL == .-TSTTY2
000256   177     000G                   ASTBL2: .BYTE   177,S2400
000260   176     000G                           .BYTE   176,S1200
000262   172     000G                           .BYTE   172,S1200
000264   171     000G                           .BYTE   171,S1200
000266   170     000G                           .BYTE   170,S1200
000270   162     000G                           .BYTE   162,S1200
                                        ;;             .BYTE   015,S600
000272   146     000G                           .BYTE   146,S300
                                        ;;             .BYTE   170,S150
000274   160     000G                           .BYTE   160,S110
000276   140     000G                           .BYTE   140,S110
000300   000     000                            .BYTE   0,0
        000024                          AB2LEN == .-ASTBL2
```

```
                                        .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  000000G 000000G         BIT     #$NOVLN,LSW2(R1);Can job use virtual lines?
000316  001060                          BNE     20$             ;Br if not
000320  032761  000000C 000000G         BIT     #<$SUCF!$LOFCF>,LSW9(R1);Doing logon/logoff processing?
000326  001054                          BNE     20$             ;Br if yes
000330  032761  000000G 000000G         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  000000G                 MOV     LNPRIM(R1),R2   ;Get primary line #
000350  016203  000000G                 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  000000G                 MOV     #FSTSL,R3       ;Get # of first virtual line
000366  020327  000000G         3$:     CMP     R3,#LSTSL       ;Checked all?
000372  101032                          BHI     20$             ;Br if yes
000374  005763  000000G                 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    $PPRED          ;Reduce execution priority of discon proc
                                ;
                                ;  Clear flag that says bell has been rung for virtual line wait
                                ;  condition for line we are switching away from.
                                ;
000514  042761  000000G 000000G         BIC     #$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
```

```
      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  0000006                       MOVB    LBSPRI(R1),R2   ;GET BASE PRIORITY FOR JOB
     11 001004  120237  0000006                       CMPB    R2,VPRILO       ;IS THIS A FIXED PRIORITY JOB?
     12 001010  101414                                BLOS    7$              ;BR IF YES
     13 001012  120237  0000006                       CMPB    R2,VPRIHI       ;REAL-TIME JOB?
     14 001016  103011                                BHIS    7$              ;BR IF YES
     15 001020  113700  0000006                       MOVB    VPRIVR,R0       ;GET AMT TO REDUCE FOR DISCONNECTED JOBS
     16 001024  160002                                SUB     R0,R2           ;REDUCE FOR DISCONNECTED JOB
     17 001026  120237  0000006                       CMPB    R2,VPRILO       ;DID PRIORITY GO OUT OF NORMAL RANGE?
     18 001032  003003                                BGT     7$              ;BR IF NOT
     19 001034  113702  0000006                       MOVB    VPRILO,R2       ;GET LOWEST NORMAL PRIORITY
     20 001040  005202                                INC     R2
     21 001042  110261  0000006              7$:      MOVB    R2,LPRI(R1)     ;SET CURRENT PRIORITY FOR JOB
     22                                       ;
     23                                       ;  Finished
     24                                       ;
     25 001046  012602                                MOV     (SP)+,R2
     26 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     URO             ;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$RTN(R1),URO  ;Return old compl rtn address in R0
001074  005062  000000G CLR     LTTCR(R2)       ;Say no completion Q element now
001100  004737  000000G CALL    QFREE           ;Release the queue element
                ;
                ;  See if the address of a new completion routine was specified
                ;
001104  013700  000020  1$:     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  000000G                 MOV     LTTCR(R1),R4    ;;;Does job want TT input compl routine?
001240  001434                          BEQ     9$              ;;;Br if not
001242  016102  000000G                 MOV     LINPNT(R1),R2   ;;;Get pointer to next char to get
001246  020261  000000G                 CMP     R2,LINNXT(R1)   ;;;Are there any pending input chars?
001252  001427                          BEQ     9$              ;;;Br if not
001254  005061  000000G                 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  000000G                 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  000000G                 MOV     R0,CQ$R0(R4)    ;Pass char to compl rtn in R0
001320  004737  000000G                 CALL    QCOMPL          ;Queue the completion routine
001324  000402                          BR      9$
                                ;
                                ;  No characters available.
                                ;  Leave the completion request pending.
                                ;
001326  010461  000000G         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
```

```
                                        .SBTTL  DOCTLT -- Display a job's state information
                                ;-----------------------------------------------------------------------
                                ;  This routine is invoked when a job types ^T.  The job does not
                                ;  have to be resident.  The format of the display is:
                                ;
                                ;       <CR>            ;in case the cursor is not at the left of the screen
                                ;       job number      ;virtual job number (may be secondary line)
                                ;       job name        ;LUNAME
                                ;       wall time       ;SYTIMH,SYTIML
                                ;       program name    ;LPRG1,LPRG2
                                ;       job size        ;LNBLKS+LNSBLK in Kb
                                ;       state           ;LSTATE
                                ;       CPU time        ;LCPUHI,LCPULO
                                ;
                                ;  Inputs:
                                ;       R1 contains job index
                                ;
001346                          DOCTLT:
001346  010246                          MOV     R2,-(SP)
001350  004737  001402'                 CALL    GTSMBF  ;Try to get a system message buffer (ptr in R2)
001354  103410                          BCS     9$      ;Just ignore ^T if no message buffer available
001356  010246                          MOV     R2,-(SP) ;Save message ptr
001360  016202  000000G                 MOV     SB$PNT(R2),R2 ;Point to text buffer
001364  004737  001572'                 CALL    BLDSTR  ;Build the output string directly into message buffer
001370  012602                          MOV     (SP)+,R2 ;Restore message ptr
001372  004737  001474'                 CALL    QCTMSG  ;Queue the ^T message for display
001376  012602                  9$:     MOV     (SP)+,R2
001400  000207                          RETURN
```

```
                                        .SBTTL  GTSMBF -- Get a system message buffer
                                ;-----------------------------------------------------------------------
                                ;  Get a free system message buffer block.  If none are available,
                                ;  return with carry set.
                                ;
                                ;  Outputs:
                                ;       R2      ptr to system message block
                                ;
001402                          GTSMBF: DISABL                  ;;; *** DISABLE ***
001410  005737  000000G                TST     NMUMB           ;;;Are there any available?
001414  001422                         BEQ     8$              ;;;If not, return with error
001416  013702  000000G                MOV     SNMSHD,R2       ;;;Point to first free buffer block
001422  001417                         BEQ     8$              ;;;If end of list, return with error
                                ;
                                ;  We have a free system message buffer.  Unlink from chain.
                                ;
001424  005337  000000G                DEC     NMUMB           ;;;One less available
001430  016237  000000G 000000G        MOV     SB$LNK(R2),SNMSHD ;;;Remove from free list
001436                                 ENABL                   ;;; *** ENABLE ***
                                ;
001444  010246                         MOV     R2,-(SP)        ;Copy buffer block ptr
001446  062716  000000G                ADD     #SB$TXT,(SP)    ;Point to text area
001452  012662  000000G                MOV     (SP)+,SB$PNT(R2) ;Init text pointer
001456  000241                         CLC                     ;No errors
001460  000404                         BR      9$              ;return
                                ;
                                ;  No free message buffers.
                                ;
001462                          8$:     ENABL                   ;;; *** ENABLE ***
001470  000261                         SEC                     ;Signal error
                                ;
001472  000207                  9$:     RETURN
```

```
                        .SBTTL  QCTMSG -- Queue ^T information to be sent to job
                ;-----------------------------------------------------------------
                ;  Inputs:
                ;       R1 index of job to receive message (same as sender)
                ;       R2 address of system message buffer
                ;
001474  010046  QCTMSG: MOV     R0,-(SP)
001476  010146          MOV     R1,-(SP)        ;Save job index, need to work on primary
                ;
001500                  DISABL                  ;;; *** DISABLE ***
001506  016101  000000G MOV     LNPRIM(R1),R1   ;;;Get physical line index
001512  016100  000000G MOV     LMSGBF(R1),R0   ;;;Get ptr to head of job's message list
001516  001411          BEQ     3$              ;;;Br if no pending messages for job
001520  005760  000000G 1$:     TST     SB$LNK(R0)      ;;;Is this the end of the list?
001524  001403          BEQ     2$              ;;;Br if yes
001526  016000  000000G MOV     SB$LNK(R0),R0   ;;;If not, point to next message
001532  000772          BR      1$              ;;;And keep searching for end
001534  010260  000000G 2$:     MOV     R2,SB$LNK(R0)   ;;;Add current message to end of list
001540  000402          BR      4$              ;;;
001542  010261  000000G 3$:     MOV     R2,LMSGBF(R1)   ;;;Say current message is first
001546  005062  000000G 4$:     CLR     SB$LNK(R2)      ;;;Say we are at end of list
001552                  ENABL                   ;;; *** ENABLE ***
                ;
                ;  Try to start transmission to the line
                ;
001560  012601          MOV     (SP)+,R1        ;Restore index of job which hit ^T
001562  004777  000000G CALL    @TRNSTR         ;Initiate transmission
                ;
001566  012600          MOV     (SP)+,R0
001570  000207          RETURN
```

```
                        .SBTTL  BLDSTR -- Build ^T information string
                ;-----------------------------------------------------------------------
                ;  Create ^T information in ASCII format into message buffer using
                ;  system table information.
                ;
                ;  Inputs:
                ;       R1 job index
                ;       R2 ptr to beginning of text buffer
                ;
001572  010046          BLDSTR: MOV     R0,-(SP)
001574  010346                  MOV     R3,-(SP)
001576  010446                  MOV     R4,-(SP)
001600  010546                  MOV     R5,-(SP)

001602  112722  000000G         MOVB    #CR,(R2)+       ;In case cursor not at left margin
                ;
                ;  Job number
                ;
001606  010100                  MOV     R1,R0           ;Get job index
001610  006200                  ASR     R0              ;Convert to job number
001612  005003                  CLR     R3              ;No filler chars needed
001614  004737  002210'          CALL    EDTDEC          ;Stuff into output buffer
001620  112722  000000G         MOVB    #SPACE,(R2)+
001624  112722  000000G         MOVB    #SPACE,(R2)+
                ;
                ;  Job name
                ;
001630  010246                  MOV     R2,-(SP)        ;Save ptr to beginning of name
001632  010103                  MOV     R1,R3           ;Copy job index
001634  070327  000006          MUL     #6.,R3          ;Convert to offset into LUNAME table
001640  062703  000000G         ADD     #LUNAME,R3      ;Convert to ptr to name of this job
001644  012704  000014          MOV     #12.,R4         ;Get max string length
001650  112300          1$:     MOVB    (R3)+,R0        ;Get next char
001652  003405                  BLE     14$             ;Terminate on 0 or 200
001654  020027  000000G         CMP     R0,#ESC         ;Disallow most punctuation (allow ESC)
001660  103401                  BLO     12$
001662  110022                  MOVB    R0,(R2)+        ;Copy printable chars to output
001664  077407          12$:    SOB     R4,1$           ;Loop up to 12 times
                ;  Now remove trailing spaces
001666  020216          14$:    CMP     R2,(SP)         ;Backed past start of name?
001670  101404                  BLOS    15$             ;Br if so
001672  124227  000000G         CMPB    -(R2),#SPACE    ;Was prev char space?
001676  001773                  BEQ     14$             ;If so, keep checking
001700  005202                  INC     R2              ;Leave last non-space char intact
001702  112722  000000G 15$:    MOVB    #SPACE,(R2)+
001706  112722  000000G         MOVB    #SPACE,(R2)+
001712  005726                  TST     (SP)+           ;Discard saved string ptr
                ;
                ;  Wall clock time
                ;
001714  013704  000000G         MOV     SYTIMH,R4       ;Get high order time (ticks)
001720  013705  000000G         MOV     SYTIML,R5       ;And low order time
001724  004737  002310'          CALL    EDTTIM          ;Stuff time into buffer
001730  112762  000000G 177776  MOVB    #SPACE,-2(R2)   ;Copy space over .
001736  112762  000000G 177777  MOVB    #SPACE,-1(R2)   ;And over tenths
                ;
                ;  Program name
```

```
                                        ;
001744  016100  000000G                 MOV     LPRG1(R1),R0    ;Get program name 1-3
001750  004737  002450'                 CALL    EDTR50          ;Stuff into output
001754  016100  000000G                 MOV     LPRG2(R1),R0    ;Get program name 4-6
001760  001402                         BEQ     2$              ;Br if less than 4 char name
001762  004737  002450'                 CALL    EDTR50
001766  126227  177776  000000G 2$:     CMPB    -2(R2),#SPACE   ;Two spaces already?
001774  001410                         BEQ     22$             ;Don't add any if so
001776  126227  177777  000000G         CMPB    -1(R2),#SPACE   ;Last char space?
002004  001402                         BEQ     21$             ;Only need 1 if so
002006  112722  000000G                 MOVB    #SPACE,(R2)+
002012  112722  000000G         21$:    MOVB    #SPACE,(R2)+
002016                         22$:
                                        ;
                                        ;  Job size
                                        ;
002016  016100  000000G                 MOV     LNBLKS(R1),R0   ;Get static job size (in pages)
002022  066100  000000G                 ADD     LNSBLK(R1),R0   ;And add PLAS region size
002026  005200                         INC     R0              ;Round up to nearest KB
002030  006200                         ASR     R0              ;Convert pages to Kb
002032  005003                         CLR     R3              ;No filler chars
002034  004737  002210'                 CALL    EDTDEC          ;Stuff size in Kb into output
002040  112722  000113                 MOVB    #'K,(R2)+       ;Append "Kb"
002044  112722  000142                 MOVB    #'b,(R2)+
002050  112722  000000G                 MOVB    #SPACE,(R2)+
002054  112722  000000G                 MOVB    #SPACE,(R2)+
                                        ;
                                        ;  Job state
                                        ;
002060  016100  000000G                 MOV     LSTATE(R1),R0   ;Get job state
002064  020027  000036                 CMP     R0,#MXSTAT      ;Last state we know about
002070  101401                         BLOS    3$              ;Br if in known state space
002072  005000                         CLR     R0              ;Else make it unknown
002074  006300                 3$:     ASL     R0              ;Convert to state index
002076  016000  002566'                 MOV     SNTABL(R0),R0   ;Get ptr to state name string
002102  112022                 32$:    MOVB    (R0)+,(R2)+     ;Copy state name string to output
002104  003376                         BGT     32$             ;0 or 200 terminates
002106  032761  000000G 000000G         BIT     #$INCOR,LSW(R1) ;In memory now?
002114  001005                         BNE     34$             ;Br if so
002116  012703  003326'                 MOV     #SWPTXT,R3      ;Job is swapped out, point to swap text
002122  005302                         DEC     R2              ;Point back to trailing NUL
002124  112322                 33$:    MOVB    (R3)+,(R2)+     ; and append
002126  003376                         BGT     33$
002130  112762  000000G 177777 34$:    MOVB    #SPACE,-1(R2)   ;Copy space over trailing nul
002136  112722  000000G                 MOVB    #SPACE,(R2)+
                                        ;
                                        ;  CPU time
                                        ;
002142  112722  000103                 MOVB    #'C,(R2)+       ;Put "CPU=" into output buffer
002146  112722  000120                 MOVB    #'P,(R2)+
002152  112722  000125                 MOVB    #'U,(R2)+
002156  112722  000075                 MOVB    #'=,(R2)+
002162  016104  000000G                 MOV     LCPUHI(R1),R4   ;Get job cpu time in ticks
002166  016105  000000G                 MOV     LCPULO(R1),R5
002172  004737  002310'                 CALL    EDTTIM          ;Move time into output buffer
                                        ;
                                        ;  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
```

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

```
                                ;
                                ;  Outputs:
                                ;       hh:mm:ss.t encoded at location pointed to by R2
                                ;       R2 is left pointing at byte immediately following tenths
                                ;
002310  010046                  EDTTIM: MOV     R0,-(SP)
002312  010346                          MOV     R3,-(SP)
002314  010446                          MOV     R4,-(SP)
002316  010546                          MOV     R5,-(SP)
                                ;
                                ;  Start by splitting ticks into two 1-word parts by dividing by
                                ;  number of ticks per minute.  This leaves two parts which will each
                                ;  fit into 1 word (24.*60.=1440.  and 60.*60.=3600.)
                                ;
002320  013703  0000006                 MOV     TK1VAL,R3       ;Get back # ticks/.1 sec
002324  070327  001130                  MUL     #600.,R3        ;Convert to # .1s/minute
002330  071403                          DIV     R3,R4           ;Split time into hrs*min and sec*ticks
                                ;
                                ;  Now convert to hours and minutes
                                ;
002332  010546                          MOV     R5,-(SP)        ;Save sec*ticks
002334  010405                          MOV     R4,R5           ;Move hrs*min for divide
002336  005004                          CLR     R4              ;Set for divide
002340  071427  000074                  DIV     #60.,R4         ;Split hours and minutes
002344  010400                          MOV     R4,R0           ;Get hours
002346  005003                          CLR     R3              ;Free format
002350  004737  002210'                 CALL    EDTDEC          ;Display hours
002354  112722  000072                  MOVB    #':,(R2)+
002360  010500                          MOV     R5,R0           ;Get minutes
002362  012703  030002                  MOV     #400*'0+2,R3    ;Format 2 digits, leading 0
002366  004737  002210'                 CALL    EDTDEC          ;Display minutes
002372  112722  000072                  MOVB    #':,(R2)+
002376  012605                          MOV     (SP)+,R5        ;Retrieve seconds and 10ths
                                ;
                                ;  Now display seconds and tenths
                                ;
002400  005004                          CLR     R4              ;Set for divide
002402  071427  000074                  DIV     #60.,R4         ;Split seconds and ticks
002406  010400                          MOV     R4,R0           ;Get seconds
002410  004737  002210'                 CALL    EDTDEC          ;Display seconds (format still in R3)
002414  112722  000056                  MOVB    #'.,(R2)+
002420  005004                          CLR     R4              ;Set for divide
002422  071437  0000006                 DIV     TK1VAL,R4       ;Split into tenths of seconds
002426  062704  000060                  ADD     #'0,R4          ;Has to be 0-9
002432  110422                          MOVB    R4,(R2)+
002434  105012                          CLRB    (R2)            ;Always terminate
                                ;
                                ;  Done!
                                ;
002436  012605                          MOV     (SP)+,R5
002440  012604                          MOV     (SP)+,R4
002442  012603                          MOV     (SP)+,R3
002444  012600                          MOV     (SP)+,R0
002446  000207                          RETURN
                                ;
                                        .SBTTL          EDTR50 -- Decode RAD50 value to ASCII
                                ;---------------------------------------------------------------
```

```
                                        ;  Inputs:
                                        ;       R0 value to be decoded
                                        ;       R2 ptr to output location
                                        ;
002450                                  EDTR50:
002450  010446                                 MOV     R4,-(SP)
002452  010546                                 MOV     R5,-(SP)
002454  010005                                 MOV     R0,R5           ;Copy value to be decoded
002456  005004                                 CLR     R4              ;Set for divide
002460  071427  003100                         DIV     #50*50,R4       ;Split out high letter
002464  116422  002516'                       MOVB    R50TBL(R4),(R2)+ ;Decode A..  to output
002470  005004                                 CLR     R4              ;Set for next divide
002472  071427  000050                         DIV     #50,R4          ;Split middle and last letters
002476  116422  002516'                       MOVB    R50TBL(R4),(R2)+ ;Decode .B.  to output
002502  116522  002516'                       MOVB    R50TBL(R5),(R2)+ ;Decode ..C to output
002506  105012                               CLRB    (R2)            ;Always terminate string
002510  012605                               MOV     (SP)+,R5
002512  012604                               MOV     (SP)+,R4
002514  000207                               RETURN
                                        .NLIST  BEX
002516      040     101     102  R50TBL:    .ASCII  / ABCDEFGHIJKLMNOPQRSTUVWXYZ$.?0123456789/
                                        .LIST   BEX
                                        .EVEN
```

```
                                        .SBTTL  STATES -- Names of job states
                                ;-----------------------------------------------------------------------
002566  003316'                 SNTABL: .WORD   50$      ;Unknown state!!!
002570  002664'                         .WORD   1$      ;S$RT
002572  002664'                         .WORD   1$              ;S$$RT
002574  002677'                         .WORD   3$      ;S$TTSC
002576  002714'                         .WORD   4$      ;S$TTFN
002600  002732'                         .WORD   5$      ;S$OTFN
002602  002752'                         .WORD   6$      ;S$HICP
002604  002770'                         .WORD   7$      ;S$TWFN
002606  003003'                         .WORD   10$     ;S$OTLO
002610  003021'                         .WORD   11$     ;S$IOFN
002612  003021'                         .WORD   11$             ;S$$HIP
002614  003037'                         .WORD   13$     ;S$CPU
002616  003047'                         .WORD   14$     ;S$LOW
002620  003047'                         .WORD   14$             ;S$$RUN
002622  003066'                         .WORD   16$     ;S$NEDQ
002624  003077'                         .WORD   17$     ;S$QMIO
002626  003110'                         .WORD   20$     ;S$QCCB
002630  003121'                         .WORD   21$     ;S$QCXB
002632  003132'                         .WORD   22$     ;S$QUSR
002634  003143'                         .WORD   23$     ;S$IOWT
002636  003153'                         .WORD   24$     ;S$OTWT
002640  003164'                         .WORD   25$     ;S$SFWT
002642  003175'                         .WORD   26$     ;S$WSMB
002644  003206'                         .WORD   27$     ;S$SPDB
002646  003217'                         .WORD   30$     ;S$INWT
002650  003227'                         .WORD   31$     ;S$QSPD
002652  003240'                         .WORD   32$     ;S$SPCB
002654  003251'                         .WORD   33$     ;S$MSWT
002656  003262'                         .WORD   34$     ;S$SPND
002660  003274'                         .WORD   35$     ;S$TMWT
002662  003305'                         .WORD   36$     ;S$WFM
        000036                  MXSTAT  = <.-SNTABL-2>/2        ;Define last state number we understand
                                ;
                                        .NLIST  BEX
002664  122 145 141             1$:     .ASCIZ  /Real-time/
002676  000                     2$:     .ASCIZ  //              ;S$$RT
002677  124 124 137             3$:     .ASCIZ  /TT_sngl_done/
002714  124 124 137             4$:     .ASCIZ  /TT_input_done/
002732  124 124 137             5$:     .ASCIZ  /TT_output_empty/
002752  111 156 164             6$:     .ASCIZ  /Interact_comp/
002770  124 151 155             7$:     .ASCIZ  /Timer_done/
003003  124 124 137             10$:    .ASCIZ  /TT_output_low/
003021  111 057 117             11$:    .ASCIZ  "I/O_complete"
003036  000                     12$:    .ASCIZ  //              ;S$$HIP
003037  103 157 155             13$:    .ASCIZ  /Compute/
003047  114 157 167             14$:    .ASCIZ  /Low_prio_comp/
003065  000                     15$:    .ASCIZ  //              ;S$$RUN
003066  127 141 151             16$:    .ASCIZ  /Wait-IOQ/
003077  127 141 151             17$:    .ASCIZ  /Wait-MIO/
003110  127 141 151             20$:    .ASCIZ  /Wait-CSH/
003121  127 141 151             21$:    .ASCIZ  /Wait-CXT/
003132  127 141 151             22$:    .ASCIZ  /Wait-USR/
003143  127 141 151             23$:    .ASCIZ  /Wait-IO/
003153  127 141 151             24$:    .ASCIZ  /Wait-TTO/
003164  127 141 151             25$:    .ASCIZ  /Wait-SHF/
```

```
     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
```

Cross reference table (CREF V05.05)

```
$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
```

Cross reference table (CREF V05.05)

```
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
```

Cross reference table (CREF V05.05)

	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		

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#						