

Table of contents

7-	1	Output list for registers
8-	1	Output list for TSX-Plus addresses
9-	1	Output list for miscellaneous fields
10-	1	Output list for line status tables
11-	1	Output list for device handler status
12-	1	Data areas
13-	1	DODUMP -- Dump entry point
14-	1	DATTIM -- Display date and time information
15-	1	DIV32 -- Divide 16-bit into 32-bit
16-	1	ERROR -- Display error status information
17-	1	REG -- Display contents of registers
18-	1	MODADR -- Display TSX.SAV module addresses
19-	1	OVRADR -- TSX-Plus overlay region
20-	1	STACK -- Display contents of stack
21-	1	MISC -- Display misc system values
22-	1	JOB -- Display info about current job
23-	1	LINE -- Display line definition tables
24-	1	DEVICE -- Display device definition tables
25-	1	OLIST -- Output a list of information
26-	1	OSTACK -- Display stack contents
27-	1	OTABLE -- Output tables of information
31-	1	OUTSTR -- Output an asciz string
32-	1	OUTVAL -- Convert and print a value
33-	1	OUTRAD -- Convert and print a RAD50 value
33-	20	BLKPAD -- Blank pad output stream
34-	1	OUTCHR -- Output an ascii character

```
1 .TITLE TSDUMP -- TSX-Plus crash dump
2 .ENABL LC
3 .DSABL CBL
4 .ENABL AMA
5 000000 .CSECT TSDUMP
6 000000 015430 .RAD50 /DMP/ ;System overlay id
```

```
7 ;-----
8 ; This module implements the TSX-Plus crash dump facility.
9 ;
10 ; Copyright (c) 1985.
11 ; S&H Computer Systems, Inc.
12 ; Nashville, Tennessee USA
13 ; All rights reserved.
14 ;
15 ; The crash dump facility provides diagnostic information when
16 ; a fatal system error occurs. It is entered with interrupts
17 ; disabled and prior initialization of several stored values.
18 ;
```

```
1 ;-----  
2 ; GLOBAL Definitions.  
3 ;  
4 . GLOBL DODUMP  
5 ;-----  
6 ; GLOBAL References.  
7 ;  
8 ;  
9 . GLOBL SS, SSEND, INTSTK, INTSND, JSTK, JSTKND  
10 . GLOBL VDMTCR, PSW  
11 . GLOBL DMPTXT, DMPOVL, DMPHND, DIEARG, DIEPC, DIESP  
12 . GLOBL TSGEN, TSTID, TSEEXEC, TSEMT, TSINIT  
13 . GLOBL TRPAR5, KPAR6, VPAR5, VPAR6  
14 . GLOBL CORUSR, NUMDEV, MAXDEV  
15 . GLOBL NPL, NSL, NDL, NIOL, NLINES, TNHL  
16 . GLOBL SYSDAT ; System date (TSGEN)  
17 . GLOBL SYTIMH, SYTIML ; System time (TSGEN)  
18 . GLOBL TK1SEC ; Number of ticks per second  
19 . GLOBL OVRADD ; Address of overlay tables
```

```

1 ;-----
2 ; MACRO Definitions.
3 ;-----
4 ;
5 ; DISABLE macro used to disable all interrupts.
6
7 .MACRO DISABL ;DISABLE INTERRUPTS
8 BIS #340,@#PSW
9 .ENDM DISABL
10
11 ;-----
12 ; STKDEF macro used to define a stack where:
13 ; BASE = base of stack
14 ; TOP = top of stack
15 ; LENGTH = length (in words) of stack
16
17 .MACRO STKDEF top, base, length
18 base: ;Base address of stack
19 .BLKW length ;Allocate stack space
20 top: ;Top address of stack
21 .ENDM STKDEF
22
23 ;-----
24 ; PRINT macro used to define a print string where
25 ; stradr = address of asciz string to print
26
27 .MACRO PRINT ?stradr
28 .NARG ARG ;Determine number of arguments
29 .IF NE, ARG ;Only if argument exists
30 MOV stradr,R1 ;Point to string address
31 .ENDC ; NE, ARG
32 CALL OUTSTR ;Display asciz string
33 .ENDM PRINT
34
35 ;-----
36 ; TTYOUT macro used to output a single character where
37 ; char = character to output
38
39 .MACRO TTYOUT ?char
40 .NARG ARG ;Determine number of arguments
41 .IF NE, ARG ;Only if argument exists
42 MOVB char,r0 ;Get output character
43 .ENDC ; NE ARG
44 CALL OUTCHR ;Display ascii character
45 .ENDM TTYOUT

```

```

1      ;-----
2      ; TBEG macro used to define a table entry print field where:
3      ; LABEL   = address label of table
4      ; NAME    = table title
5      ; ROWNAM  = row name label
6      ; ATTR   = index number attribute
7      ; SINDX  = starting index
8      ; EINDX  = ending index
9      ;
10     .MACRO TBEG    label, name, rownam, attr, sindx, eindx
11     .NARG  ARG5    ;Number of arguments to macro call
12     .PSECT DUMP    ;Define DUMP program section
13     label  =      ;Define starting address
14     .PSECT TSDUMP ;Restore TSDUMP program section
15     .PSECT STRING ;Define TEXT program section
16     STRADD =      ;Point to the starting address
17     .ASCII /name/ ;Store the ascii label
18     .BYTE  0      ;Terminate as asciz string
19     .PSECT DUMP    ;Define DUMP program section
20     .WORD  STRADD  ;Store the string address
21     .IF    EQ, ARG5-2 ;Only two arguments passed
22         TENABL = 0 ;Flag list definition in progress
23     .MEXIT ;Terminate macro expansion
24     .ENDC ; EQ, ARG5-2
25     TENABL = 1 ;Flag table definition in progress
26     .PSECT STRING ;Define STRING program section
27     STRADD =      ;Point to the starting address
28     .ASCII /rownam/ ;Store the ascii label
29     .BYTE  200    ;Terminate as asciz string
30     .PSECT DUMP    ;Define DUMP program section
31
32     .WORD  STRADD  ;Store the string address
33     .WORD  attr    ;Store table attributes
34     .WORD  STRADD  ;Store the string address
35     .WORD  sindx   ;Store starting table index
36     .WORD  eindx   ;Store ending table index
37     .PSECT TSDUMP ;Restore TSDUMP program section
38     .ENDM TBEG

```

```

1
2 ; -----
3 ; PVAL macro used to define a single print field where:
4 ;   NAME      = ascii string title
5 ;   LABEL     = address label containing print value
6 ;   ATTR      = field attributes
7 ;   SINX      = starting table index
8 ;   EINX      = ending table index
9 ;   Format attributes:
10 ;   $RJUST    = (bit 14) right justify (default is left)
11 ;   $ZRPAD    = (bit 13) zero pad (default is blank)
12 ;   Storage attributes:
13 ;   $BYTE     = (bit 0) storage requirement (default is word)
14 ;   $DEC      = (bit 2) decimal representation (default is octal)
15 ;   $RAD50    = (bit 3) rad50 representation
16 ;
17 ; Definition of field attribute bit definitions.
18
19         040000      $RJUST = 40000      ;Right justify
20         020000      $ZRPAD = 20000      ;Zero pad
21 ; Storage attributes.
22         000001      $BYTE  = 1          ;Byte storage cell
23         000002      $DEC   = 2          ;Decimal numeric representation
24         000004      $RAD50 = 4          ;Rad50 representation
25
26         .MACRO PVAL name, label, attr, sindx, eindx
27         .PSECT STRING
28             STRADD =
29             .ASCII /name/
30             .BYTE 200
31         .PSECT DUMP
32             .WORD STRADD
33             .WORD attr
34             .GLOBL label
35             .WORD label
36             .IF NE, tenabl
37                 .WORD sindx
38                 .WORD eindx
39             .ENDC
40         .PSECT TSDUMP
41         .ENDM PVAL

```

1  
2  
3  
4  
5  
6  
7  
8  
9

```
-----  
; TEND macro used to define the end of a table or list.  
;  
;  
; .MACRO TEND  
; .PSECT DUMP ; Define DUMP program section  
; .WORD 0 ; End of table or list  
; .PSECT TSDUMP ; Restore TSDUMP program section  
; .ENDM TEND
```

1  
2  
3  
4  
5 000002  
6 000002  
7 000002  
8 000002  
9 000002  
10 000002  
11 000002  
12 000002  
13 000002  
14 000002  
15 000002  
16 000002  
17

.SBTTL Output list for registers

-----  
; Generate list for output of register contents.  
;

TREG PRTREG, <Register Contents> ;Beginning of print list  
PVAL <R0>, SVR0 ;Register 0  
PVAL <R1>, SVR1 ;Register 1  
PVAL <R2>, SVR2 ;Register 2  
PVAL <R3>, SVR3 ;Register 3  
PVAL <R4>, SVR4 ;Register 4  
PVAL <R5>, SVR5 ;Register 5  
PVAL <SP>, DIESP ;Stack pointer  
PVAL <PC>, DIEPC ;Program counter  
PVAL <KPAR 5>, SVPAR5 ;Kernal PAR 5  
PVAL <KPAR 6>, SVPAR6 ;Kernal PAR 6  
TEND ;End of print list

```
1  
2  
3  
4  
5 000002  
6 000102  
7 000002  
8 000002  
9 000002  
10 000002  
11 000002  
12 000002  
13 000002  
14 000002
```

.SBTTL Output list for TSX-Plus addresses  
-----  
; Generate list for output of TSX-Plus address values.  
;  
TBEQ ADDR, <TSX-Plus module addresses>  
PVAL <TSGEN >,ADGEN ;Address of TSGEN  
PVAL <TSTIO >,ADTIO ;Address of TSTIO  
PVAL <TSEXEC>,ADEXEC ;Address of TSEXEC  
PVAL <TSEMT >,ADEMT ;Address of TSEMT  
PVAL <TSINIT>,ADINIT ;Address of TSINIT  
PVAL <TSTIOX>,ADTIOX ;Address of TSTIOX  
PVAL <TSLOCK>,ADLOCK ;Address of TSLOCK  
PVAL <TSCASH>,ADCASH ;Address of TSCASH  
TEND

Output list for miscellaneous fields

```

1
2
3
4
5 000002          .SBTTL  Output list for miscellaneous fields
6 000166          ;-----
7 000002          ; Generate list for output of miscellaneous fields.
8 000002          ;
9 000002          ;
10 000002         TBEG      VMISC, <Miscellaneous values> ;Beginning of print list
11 000002         PVAL      <CLKPC>, CLKPC           ;Clock interrupted PC (for PM)
12 000002         PVAL      <CLKPS>, CLKPS           ;Clock interrupted PS (for PM)
13 000002         PVAL      <CSHBAS>, CSHBAS          ;Physical add of TSCASH
14 000002         PVAL      <CURFRK>, CURFRK          ;Address of current FORK call
15 000002         PVAL      <CXBJOB>, CXBJOB, <#BYTE> ;Context block job contents
16 000002         PVAL      <CXBDWN>, CXBDWN, <#BYTE> ;Context block job owner
17 000002         PVAL      <DOSCHD>, DOSCHD, <#BYTE> ;Do scheduling flag
18 000002         PVAL      <EXCJOB>, EXCJOB, <#BYTE> ;Exclusive system job owner
19 000002         PVAL      <FREIOQ>, FREIOQ          ;Free I/O queue head
20 000002         PVAL      <FRKCQE>, FRKCQE          ;Fork queue head
21 000002         PVAL      <FREFRK>, FREFRK          ;Free fork queue head
22 000002         PVAL      <FRKPRI>, FRKPRI, <#BYTE> ;Fork priority
23 000002         PVAL      <INBSY>, INBSY, <#BYTE>   ;Inswap flag
24 000002         PVAL      <INTLVL>, INTLVL, <#BYTE> ;Interrupt level
25 000002         PVAL      <INTPRI>, INTPRI          ;Interrupt priority
26 000002         PVAL      <IOABFL>, IOABFL          ;I/O abort flag (0=rundown; 1=abort)
27 000002         PVAL      <LOKBAS>, LOKBAS          ;PAR base of TSLOCK
28 000002         PVAL      <LOKSWP>, LOKSWP          ;Job requiring memory lock
29 000002         PVAL      <MAPUSR>, MAPUSR, <#BYTE>   ;Job currently mapped
30 000002         PVAL      <MEM256>, MEM256, <#BYTE> ;Extended memory flag
31 000002         PVAL      <MEMSWP>, MEMSWP          ;Job requiring memory expansion
32 000002         PVAL      <MIOFLO>, MIOFLO, <#BYTE> ;I/O mapping flag
33 000002         PVAL      <MSGBAS>, MSGBAS          ;PAR base of TSMSG
34 000002         PVAL      <NMFREQ>, NMFREQ          ;Number of free user's queue elements
35 000002         PVAL      <NUMON>, NUMON, <#BYTE>   ;Num of hardware lines on
36 000002         PVAL      <OUTBSY>, OUTBSY, <#BYTE> ;Outswap flag
37 000002         PVAL      <PVON>, PVON, <#BYTE>     ;Num of primary and virtual lines on
38 000002         PVAL      <SPDJOB>, SPDJOB, <#BYTE> ;SPD job owner
39 000002         PVAL      <SR3FLG>, SR3FLG, <#BYTE> ;Extended memory management reg. flag
40 000002         PVAL      <STKLVL>, STKLVL, <#BYTE> ;Stack level
41 000002         PVAL      <TIOBAS>, TIOBAS          ;Physical add of TSTIOX
42 000002         PVAL      <TOTON>, TOTON, <#BYTE>   ;Num of total lines on
43 000002         PVAL      <UBUSMP>, UBUSMP, <#BYTE> ;Unibus map flag
44 000002         PVAL      <UIOCNT>, UIOCNT          ;User I/O count
45 000002         PVAL      <USP>, USP                ;User stack pointer
46 000002         PVAL      <USRBAS>, USRBAS          ;PAR base of TSUSR
47 000002         PVAL      <USRJOB>, USRJOB, <#BYTE> ;USR job owner
48 000002         PVAL      <VBUSTP>, VBUSTP, <#BYTE> ;Bus type (0 = UNIBUS; 1 = QBUS)
49 000002         PVAL      <VCSHNB>, VCSHNB          ;Num of generalized cache blocks
50 000002         PVAL      <VMXSF>, VMXSF           ;Max num of shared files
51 000002         PVAL      <VMXSFC>, VMXSFC          ;Max num of shared file channels
52 000002         PVAL      <VNGR>, VNGR            ;Num of global PLAS regions
53 000002         PVAL      <VNUMDC>, VNUMDC          ;Num of shared data cache blocks
54 000002         PVAL      <VPLAS>, VPLAS           ;Num of PLAS blocks allocated
55 000002         PVAL      <VSWPFL>, VSWPFL, <#BYTE> ;Swap flag (0 = swap; 1 = noswap)
56 000002         PVAL      <VSWPSL>, VSWPSL          ;Num of swap file jobs allocated
57 000002         PVAL      <VUXIFL>, VUXIFL, <#BYTE> ;Unexpected interrupt (1 = abort)
58 000002         TEND

```

Output list for line status tables

```

1
2
3
4
5 000002          .SBTTL  Output list for line status tables
6 000002          ;-----
7 000002          ;  Generate table for output of line status.
8 000002          ;
9 000002          TBEG  JLIN, <Job Status Tables>, <Job>, <#BYTE!#DEC>, 1, NLINES
10 000002         PVAL  <LSW>, LSW, <#RJUST!#ZRPAD>      ; Job status word
11 000002         PVAL  <LSW2>, LSW2, <#RJUST!#ZRPAD>    ; Job status word 2
12 000002         PVAL  <LSW3>, LSW3, <#RJUST!#ZRPAD>    ; Job status word 3
13 000002         PVAL  <LSW4>, LSW4, <#RJUST!#ZRPAD>    ; Job status word 4
14 000002         PVAL  <LSW5>, LSW5, <#RJUST!#ZRPAD>    ; Job status word 5
15 000002         PVAL  <LSW6>, LSW6, <#RJUST!#ZRPAD>    ; Job status word 6
16 000002         PVAL  <LSW7>, LSW7, <#RJUST!#ZRPAD>    ; Job status word 7
17 000002         PVAL  <LSW8>, LSW8, <#RJUST!#ZRPAD>    ; Job status word 8
18 000002         PVAL  <LSW9>, LSW9, <#RJUST!#ZRPAD>    ; Job status word 9
19 000002         PVAL  <LSW10>, LSW10, <#RJUST!#ZRPAD>  ; Job status word 10
20 000002         PVAL  <LSW11>, LSW11, <#RJUST!#ZRPAD>  ; Job status word 11
21 000002         PVAL  <LSTATE>, LSTATE, <#RJUST!#ZRPAD> ; Job state
22 000002         PVAL  <LBASE>, LBASE, <#RJUST!#ZRPAD>  ; Base 512-block num of context
23 000002         PVAL  <LPARBS>, LPARBS, <#RJUST!#ZRPAD> ; Base PAR of program
24 000002         PVAL  <LNBLKS>, LNBLKS, <#RJUST!#ZRPAD> ; Num of mem pages for job
25 000002         PVAL  <LNSBLK>, LNSBLK, <#RJUST!#ZRPAD> ; Num of mem pages for PLAS
26 000002         PVAL  <LIOCNT>, LIOCNT, <#RJUST!#ZRPAD> ; Active I/O counter
27 000002         PVAL  <LINCNT>, LINCNT, <#RJUST!#ZRPAD> ; Input chars pending
28 000002         PVAL  <LACTIV>, LACTIV, <#RJUST!#ZRPAD> ; Activ chars pending
29 000002         PVAL  <LPRG1>, LPRG1, <#RAD50>          ; Program name
30 000002         PVAL  <LPRG2>, LPRG2, <#RAD50>          ; Program name
31 000002         PVAL  <LCDTYP>, LCDTYP, <#RJUST!#ZRPAD>, 1, TNHL ; Hardware line status
32 000002         PVAL  <LNPRIM>, LNPRIM, <#RJUST!#ZRPAD>, 1, NPL+NDL; Primary line index
33 000002         PVAL  <LPARNT>, LPARNT, <#RJUST!#ZRPAD>  ; Parent number index
34 000002         PVAL  <LBSPRI>, LBSPRI, <#RJUST!#ZRPAD>  ; Job base priority
35 000002         PVAL  <LCLUNT>, LCLUNT, <#RJUST!#ZRPAD>, 1, NPL+NSL+NDL+NIOL; CL unit
36 000002         PVAL  <LMEMIN>, LMEMIN, <#RJUST!#ZRPAD>  ; Num mem pages require
37 000002         PVAL  <LIOHLD>, LIOHLD, <#RJUST!#ZRPAD>  ; I/O hold for swap
38 000002         PVAL  <LAFSIZ>, LAFSIZ, <#RJUST!#ZRPAD>  ; Field width activate
39 000002         PVAL  <LFWLIM>, LFWLIM, <#RJUST!#ZRPAD>  ; Field limit activate
40 000002         PVAL  <LCMPL>, LCMPL, <#RJUST!#ZRPAD>    ; Pending completion
41 000002         PVAL  <LSWPBK>, LSWPBK, <#RJUST!#ZRPAD>  ; Swap file block num
42 000002         PVAL  <LJSW>, LJSW, <#RJUST!#ZRPAD>      ; Job status word
43 000002         PVAL  <LEMTPC>, LEMTPC, <#RJUST!#ZRPAD>  ; Last EMT PC address
44 000002         PVAL  <LSPND>, LSPND, <#RJUST!#ZRPAD>    ; Job .SPND counter
45 000002         PVAL  <LXCL>, LXCL, <#RJUST!#ZRPAD>, 1, NPL; Cross connect CL unit index
46 000002         TEND

```

1  
2  
3  
4  
5 000002  
6 000002  
7 000002  
8 000002  
9 000002  
10 000002  
11 000002  
12 000002  
13 000002  
14 000002

.SBTTL Output list for device handler status

-----  
; Generate table for output of device handler status.  
;

TBEG DHAN, <Device Handler Tables>, <Offset>, <#BYTE!#DEC>, 0, MAXDEV-1  
PVAL <PNAME>, PNAME, <#RAD50> ; Program name  
PVAL <HANENT>, HANENT, <#RJUST> ; Handler entry point  
PVAL <HANPAR>, HANPAR, <#RJUST> ; Handler PAR  
PVAL <HANSIZ>, HANSIZ, <#RJUST> ; Handler size  
PVAL <HANIOC>, HANIOC, <#RJUST> ; Handler I/O count  
PVAL <DVFLAG>, DVFLAG, <#RJUST!#ZRPAD> ; Device handler flags  
PVAL <DVSTAT>, DVSTAT, <#RJUST!#ZRPAD> ; Device status flags  
PVAL <DEVSIZ>, DEVSIZ, <#RJUST!#ZRPAD> ; Device size (256 word blks)  
TEND

Data areas

```

1          .SBTTL  Data areas
2          ;-----
3          ; Internal storage.
4          ;
5          ; Assignments.
6          ;
7          000015 CR      =      15      ; Carriage return
8          000012 LF      =      12      ; Line feed
9          000021 CTRLQ   =      21      ; Control-Q
10         000023 CTRLS   =      23      ; Control-S
11         000055 DASH    =      '-'     ; Dash
12         000014 NCOL    =      12      ; Number of columns for table output
13
14         ;
15         ; Internal stack and register contents.
16         ;
17         000002 000000 SVR0:  .WORD  0
18         000004 000000 SVR1:  .WORD  0
19         000006 000000 SVR2:  .WORD  0
20         000010 000000 SVR3:  .WORD  0
21         000012 000000 SVR4:  .WORD  0
22         000014 000000 SVR5:  .WORD  0      ; Saved registers (r5 to r0)
23         000016 000000 SVREG  =      ; Past saved registers
24         000016 000000 SVPAR5: .WORD  0      ; Saved kernel PAR 5
25         000020 000000 SVPAR6: .WORD  0      ; Saved kernel PAR 6
26
27         ; Root module addresses.
28         ;
29         000022 000000 ADGEN:  .WORD  0      ; Address of TSGEN
30         000024 000000 ADTIO:  .WORD  0      ; Address of TSTIO
31         000026 000000 ADEXEC: .WORD  0      ; Address of TSEEXEC
32         000030 000000 ADEMT:  .WORD  0      ; Address of TSEMT
33         000032 000000 ADINIT: .WORD  0      ; Address of TSINIT
34         000034 000000 ADTIOX: .WORD  0      ; Address of TSTIOX
35         000036 000000 ADLOCK: .WORD  0      ; Address of TSLOCK
36         000040 000000 ADCASH: .WORD  0      ; Address of TSCASH
37
38         ; Miscellaneous storage.
39         ;
40         ; Word bounded storage.
41         ;
42         000042 000000 TBUF:  .WORD  0      ; Transmit buffer
43         000044 000000 TCSR:  .WORD  0      ; Transmit CSR
44         000046 000000 RBUF:  .WORD  0      ; Receiver buffer
45         000050 000000 RCSR:  .WORD  0      ; Receiver CSR
46         000052 000000 COL:   .WORD  0      ; Current column position
47
48         ; Format tables for columnar output.
49         ;
50         000054 TSCOL:  .BLKW  NCOL+1      ; Starting column on table output
51         000106 TECOL:  .BLKW  NCOL+1      ; Ending column on table output
52         000140 TSINX:  .BLKW  NCOL+1      ; Starting index for column
53         000172 TEINX:  .BLKW  NCOL+1      ; Ending index for column
54         000224 TATT:   .BLKW  NCOL+1      ; Attributes of table
55         000256 TADR:   .BLKW  NCOL+1      ; Base address of table
56
57         ; Data.

```

Data areas

```

58 ;
59 000310 037266 023112 050572 MONTH: .RAD50 /JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC/
    000316 004322 050601 040726
    000324 040724 004617 073630
    000332 057114 054756 014713
60 ;
61 ; Byte bounded storage.
62 ;
63 000340 021 XFLG: .BYTE CTRLQ ;XON/XOFF flag
64 ;
65 ; Output strings.
66 ;
67 .NLIST BEX
68 000341 101 162 147 TXARG: .ASCII /Arg. value = /<200>
69 000357 123 145 147 TXSEG: .ASCII /Seg. value = /<200>
70 000375 117 166 145 TXOID: .ASCII /Overlay: /<200>
71 000407 104 145 166 TXDEV: .ASCII /Device name: /<200>
72 000425 040 075 040 EQSTR: .ASCII / = /<200>
73 000431 040 040 040 SPSTR: .ASCII / /<200>
74 000437 014 200 FF: .BYTE 14, 200 ;Form feed
75 000441 015 012 200 CRLF: .BYTE CR, LF, 200 ;Carriage return / line feed
76 000444 124 123 130 OVLREG: .ASCIZ /TSX-Plus overlay regions/
77 000475 123 171 163 STKSS: .ASCIZ /System /<200>
78 000506 111 156 164 STKINT: .ASCIZ /Interrupt /<200>
79 000522 105 115 124 STKEMT: .ASCIZ /EMT /<200>
80 000530 123 164 141 STKCNT: .ASCIZ /Stack Contents/
81 000547 040 057 040 STKDLM: .ASCII \ / \<200>
82 000553 111 156 166 INVSTK: .ASCIZ /Invalid stack pointer/
83 000601 111 156 164 INTLEV: .ASCIZ /Interrupt Level/
84 000621 106 157 162 FRKLEV: .ASCIZ /Fork Level/
85 000634 123 171 163 STKOV: .ASCIZ /System stack overflow/
86 000662 112 157 142 CURJOB: .ASCII /Job executing at time of dump/<200>
87 .LIST BEX
88 .EVEN

```

```

1          .SBTTL  DODUMP -- Dump entry point
2          ;-----
3          ; TSDUMP entry point.  TSDUMP is entered from TSEXEC with the
4          ; following known values:
5          ;     DMPTXT  = Asciz text of die message
6          ;     DMPOVL  = Rad50 overlay name (or zero)
7          ;     DMPHND  = Rad50 handler name (or zero)
8          ;     DIEMSG  = Address of die message
9          ;     DIEARG  = Argument of die message
10         ;     DIESP   = Original stack pointer at time of crash
11         ;     DIEPC   = Program counter of DIE call
12         ;     TRPAR5  = PAR 5 value at trap
13         ;     SP      = All registers - R0,R1,R2,R3,R4,R5
14         ;
15         ; A non-interrupt output routine is used to display information to
16         ; an XDN/XOFF device or a parallel port printer.  The following
17         ; information is output:
18         ;     Registers (R0 through R5)
19         ;     SP (stack pointer)
20         ;     PC (program counter from error)
21         ;     PSW
22         ;     Stack contents
23         ;     Miscellaneous data cells
24         ;     Line definition tables
25         ;     Device definition tables
26         ;
27 000720  DODUMP:
28         ;
29         ; Disable interrupt and locate the dump device registers.
30         ;
31         ;     DISABL          ; Disable all interrupts
32 000720  012737  001124' 000004  MOV     #DMPTRP,@#4    ; Capture the trap 4 address
33         ;     MOV     #340,@#6    ; Retain disable interrupt status
34 000726  013705  0000000  MOV     VDMTCR,R5     ; Find the display device CSR
35 000732  042705  000004    BIC     #4,R5         ; Adjust for receiver CSR
36 000736  005015    CLR     (R5)         ; Check for receiver CSR
37 000740  103411    BCS    1$           ; Br if receiver not available
38 000742  010537  000050'  MOV     R5,RCSR      ; Insert receiver address
39 000746  010537  000046'  MOV     R5,RBUF      ; Insert receiver buffer
40 000752  062737  000002  000046'  ADD     #2,RBUF      ; offset 2 from CSR
41 000760  117700  177062    MOVB   @RBUF,R0     ; Throw away any pending input
42 000764  062705  000004    1$:    ADD     #4,R5     ; Adjust to transmit CSR
43 000770  005015    CLR     (R5)         ; Check for transmitter CSR
44 000772  103453    BCS    50$          ; Br if transmitter not available
45 000774  010537  000044'  MOV     R5,TCSR      ; Insert transmitter CSR
46 001000  010537  000042'  MOV     R5,TBUF      ; Insert transmitter buffer address
47 001004  062737  000002  000042'  ADD     #2,TBUF      ; offset 2 from CSR
48         ;
49         ; Save values on entry (registers, kernel mapping, etc.).
50         ;
51 001012  013737  0000000 000020'  MOV     @#KPAR6,SVPAR6 ; Save kernel PAR 6
52 001020  013737  0000000 000016'  MOV     TRPAR5,SVPAR5  ; Save kernel PAR 5
53 001026  012705  000016'  MOV     #SVREG,R5     ; Save registers on current stack
54 001032  012700  000006    MOV     #6,R0        ; Save registers R0-R6
55 001036  012645    10$:    MOV     (SP)+,-(R5)   ; Save all the register contents
56 001040  077002    SOB    R0,10$        ; Continue until all register saved
57

```

```
58      ;  
59      ;    Output information concerning the current state.  
60      ;  
61 001042      PRINT    #FF                    ; Output form feed character  
62 001052    004737    001134'    CALL    DATTIM            ; Output date and time information  
63 001056    004737    001462'    CALL    ERROR            ; Output error status information  
64 001062    004737    002400'    CALL    JOB              ; Output current job information  
65 001066    004737    001626'    CALL    REG              ; Output the register contents  
66 001072    004737    001652'    CALL    MODADR           ; Output TSX-Plus module addresses  
67 001076    004737    001776'    CALL    DVRADR           ; Output TSX-Plus overlay regions  
68 001102    004737    002300'    CALL    MISC             ; Output miscellaneous information  
69 001106    004737    002460'    CALL    LINE             ; Output line table information  
70 001112    004737    002472'    CALL    DEVICE           ; Output handler table information  
71 001116    004737    002072'    CALL    STACK            ; Output the stack contents  
72 001122    000000                    50#:    HALT              ; Now where to?  
73      ;  
74      ;    Trap to 4 control.  
75      ;  
76 001124    052766    000001    000002    DMPTRP: BIS        #1,2(SP)        ; Buffer not found, set c-bit  
77 001132    000002                    RTI                    ; Return from trap
```

```

1          .SBTTL  DATTIM -- Display date and time information
2          ;-----
3          ; DATTIM is called to display the current date and time information.
4          ;
5 001134   DATTIM: PRINT  #CRLF          ;Display CR/LF
6 001144   013705 0000000  MOV      SYSDAT,R5      ;Get current date
7 001150   001451          BEQ      10$          ;No date set - skip date
8 001152   010501          MOV      R5,R1          ;Copy to working register
9 001154   072127 177773  ASH      #-5,R1          ;Find the day (bits 5-9)
10 001160  042701 177740  BIC      #^C37,R1       ;Mask unwanted bits
11 001164  012703 000002  MOV      ##DEC,R3       ;Force decimal display
12 001170  004737 003626' CALL     OUTVAL          ;Print the day
13 001174          TTYOUT  #'-          ;Output "-"
14 001204  010501          MOV      R5,R1          ;Copy date to working register
15 001206  072127 177767  ASH      #-9.,R1        ;Find the month (bits 10-13)
16 001212  042701 177741  BIC      #^C36,R1       ;Mask unwanted bits (word offset)
17 001216  016101 000306' MOV      MONTH-2(R1),R1  ;Offset into month table
18 001222  012703 000004  MOV      ##RAD50,R3     ;Force rad50 display
19 001226  004737 004022' CALL     OUTRAD          ;Output three char month
20 001232          TTYOUT  #'-          ;Output "-"
21 001242  010501          MOV      R5,R1          ;Copy date to working register
22 001244  042701 177740  BIC      #^C37,R1       ;Find the year (bits 0-4)
23 001250  062701 000110  ADD      #72.,R1        ;Year is relative to 1972
24 001254  012703 000002  MOV      ##DEC,R3       ;Force decimal display
25 001260  004737 003626' CALL     OUTVAL          ;Print blank space
26 001264          PRINT  #SPSTR          ;Print blank space
27          ;
28          ; Output the current time.
29          ;
30 001274  013700 0000000 10$:    MOV      SYTIMH,R0      ;Get the current time (high)
31 001300  013701 0000000  MOV      SYTIML,R1      ; clock ticks past midnight (low)
32 001304  013703 0000000  MOV      TK1SEC,R3      ;Get number of clock ticks per second
33 001310  004737 001424' CALL     DIV32          ;Divide clock ticks per second
34 001314  010246          MOV      R2,-(SP)        ;Save ticks
35 001316  012703 000074  MOV      #60.,R3        ;Get 60. per unit
36 001322  012705 000002  MOV      #2,R5          ;Process seconds, minutes, hours
37 001326  004737 001424'11$:    CALL     DIV32          ;Divide by 60. per unit
38 001332  010246          MOV      R2,-(SP)        ;Save remainder
39 001334  077504          SOB      R5,11$         ;Continue for seconds, minutes, hours
40 001336  010146          MOV      R1,-(SP)        ;Save final quotient
41 001340  012705 000004  MOV      #4,R5          ;Loop for minutes, seconds, and ticks
42 001344  000404          BR      21$            ;Output the time past midnight
43 001346          20$:    TTYOUT  #':'          ;Output ":"
44 001356  012601          21$:    MOV      (SP)+,R1      ;Get the next value
45 001360  005000          CLR      R0            ;Clear high order
46 001362  071027 000012  DIV      #10.,R0        ;Divide by ten
47 001366  062700 000060  ADD      #'0,R0         ;Convert to ascii number
48 001372          TTYOUT  R1            ;Display high digit (quotient)
49 001376  062701 000060  ADD      #'0,R1         ;Convert to ascii number
50 001402          TTYOUT  R1            ;Display low digit (remainder)
51 001410  077522          SOB      R5,20$         ;Continue for hours, minutes, seconds, ticks
52 001412          PRINT  #CRLF          ;Display CR/LF
53 001422  000207          RETURN

```

DIV32 -- Divide 16-bit into 32-bit

```

1
2
3
4
5
6
7
8 001424 010446
9 001426 005002
10 001430 012704 000037
11 001434 006301
12 001436 006100
13 001440 006102
14 001442 020203
15 001444 103402
16 001446 160302
17 001450 005201
18 001452 005304
19 001454 100367
20 001456 012604
21 001460 000207

```

```

.SBTTL DIV32 -- Divide 16-bit into 32-bit
-----
; DIV32 divides a 16-bit number in R3 into a 32-bit number
; in R0 (high order) and R1 (low order). The quotient is
; returned in R0, R1 and the remainder is returned in R2.
; All other registers are preserved.
;
DIV32: MOV R4, -(SP) ; Save registers
      CLR R2 ; Initialize remainder
      MOV #31, R4 ; Initialize shift count
1$: ASL R1 ; Shift from low order
   ROL R0 ; through high order
   ROL R2 ; into remainder
   CMP R2, R3 ; Check remainder for subtraction
   BLO 2$ ; Br if unable to subtract
   SUB R3, R2 ; Subtract divisor
   INC R1 ; Increment quotient
2$: DEC R4 ; Check remaining bit shift count
   BPL 1$ ; Continue if more to consider
   MOV (SP)+, R4 ; Restore registers
   RETURN

```

ERROR -- Display error status information

```

1          .SBTTL  ERROR  -- Display error status information
2          ;-----
3          ;  ERKOR is called to display the error status information.
4          ;
5 001462  ERROR:  PRINT  #CRLF          ; Display CR/LF
6 001472          PRINT  #DMPTXT       ; Display fatal error message
7 001502          PRINT  #TXARG        ; Display "Arg. value"
8 001512  013701  0000000  MOV  DIEARG,R1  ; Get the argument value
9 001516  005003          CLR  R3       ; Set display characteristics
10 001520  004737  003626'  CALL  OUTVAL    ; Output argument value
11 001524          PRINT  #CRLF        ; Display CR/LF
12 001534  013705  0000000  MOV  DMPOVL,R5  ; Get the rad50 overlay identifier
13 001540  001413          BEQ  1$     ; Br if zero, not in overlay
14 001542          PRINT  #TXOID       ; Display "Overlay: "
15 001552  010501          MOV  R5,R1   ; Copy overlay identifier
16 001554  004737  004022'  CALL  OUTRAD    ; Output rad50 overlay identifier
17 001560          PRINT  #CRLF        ; Display CR/LF
18 001570  013705  0000000  1$:  MOV  DMPHND,R5  ; Get the rad50 handler identifier
19 001574  001413          BEQ  2$     ; Br is zero, not in handler
20 001576          PRINT  #TXDEV       ; Display "Device name: "
21 001606  010501          MOV  R5,R1   ; Copy device handler identifier
22 001610  004737  004022'  CALL  OUTRAD    ; Output rad50 handler identifier
23 001614          PRINT  #CRLF        ; Display CR/LF
24 001624  000207          2$:  RETURN  ; Finished
25

```

REG -- Display contents of registers

```
1 .SBTTL REG -- Display contents of registers
2 ;-----
3 ; REG is called to display the contents of the registers.
4 ;
5 001626 REG: PRINT #CRLF ;Display CR/LF
6 001636 005002 CLR R2 ;Clear offset pointer
7 001640 012705 000000' MOV #PRTREG,R5 ;Point to register output table
8 001644 004737 002504' CALL OLIST ;Output the list of values
9 001650 000207 RETURN
```

MODADR -- Display TSX.SAV module addresses

```

1          .SBTTL  MODADR  -- Display TSX.SAV module addresses
2          ;-----
3          ; MODADR is called to display the module address contained in TSX.SAV.
4          ;
5 001652   MODADR:
6 001652   PRINT    #CRLF          ;Display CR/LF
7 001662   012737   0000000 000022'  MOV    #TSGEN,ADGEN  ;Save the TSGEN address
8 001670   012737   0000000 000024'  MOV    #TSTIO,ADTIO  ;Save the TSTIO address
9 001676   012737   0000000 000026'  MOV    #TSEXEC,ADEXEC ;Save the TSEXEC address
10 001704   012737   0000000 000030'  MOV    #TSEMT,ADEMT  ;Save the TSEMT address
11 001712   012737   0000000 000032'  MOV    #TSINIT,ADINIT ;Save the TSINIT address
12 001720   013705   0000000          MOV    TIOBAS,R5     ;Get PAR of TSTIOX
13 001724   072527   0000006          ASH    #6,R5         ;Convert to address
14 001730   010537   0000034'        MOV    R5,ADTIOX     ;Get the TSTIOX address
15 001734   013705   0000000          MOV    LOKBAS,R5     ;Save PAR of TSLOCK
16 001740   072527   0000006          ASH    #6,R5         ;Convert to address
17 001744   010537   0000036'        MOV    R5,ADLOCK     ;Save the TSLOCK address
18 001750   013705   0000000          MOV    CSHBAS,R5     ;Get PAR of TSCASH
19 001754   072527   0000006          ASH    #6,R5         ;Convert to address
20 001760   010537   0000040'        MOV    R5,ADCASH     ;Save the TSCASH address
21 001764   012705   000100'         MOV    #ADDR,R5      ;Point to address output list
22 001770   004737   002504'         CALL   OLIST         ;Output address contents
23 001774   000207          RETURN

```

OVRADR -- TSX-Plus overlay region

```

1          .SBTTL  OVRADR -- TSX-Plus overlay region
2          ;-----
3          ; OVRADR is called to display the overlay region locations.
4          ;
5          ; Overlay table structure:
6          ;
7          ; OVRADD --> #OVTAB:
8          ;          WORD  <IDENTIFIER>, <KPAR5>, <WORD COUNT>
9          ;          DUMMY SUBROUTINES FOR ALL OVERLAY SEGMENTS
10         ;
11 001776  OVRADR:
12 001776          PRINT  #CRLF          ;Display CR/LF
13 002006          PRINT  #OVLREG        ;Display header
14 002016  013705  0000000          MOV    OVRADD,R5          ;Find the overlay table address
15 002022  012501          10$:  MOV    (R5)+,R1          ;Get the overlay identifier
16 002024  004737  004022'        CALL  OUTRAD          ;Display rad50 id
17 002030          PRINT  #EQSTR        ;Display " = "
18 002040  012501          MOV    (R5)+,R1          ;Get the KPAR5 value
19 002042  005003          CLR    R3          ;Default octal display value
20 002044  004737  003626'        CALL  OUTVAL          ;Display KPAR5 value
21 002050  005725          TST    (R5)+          ;
22 002052          PRINT  #CRLF        ;Display CR/LF
23 002062  021527  004537          CMP    (R5),#4537          ;Compare with a <JSR R5,#OVRH> instruction
24 002066  001355          BNE    10$          ;Br if not end of the table
25 002070  000207          RETURN

```

STACK -- Display contents of stack

```

1          .SBTTL  STACK  -- Display contents of stack
2          ;-----
3          ; STACK is called to display the contents of the stack.
4          ;
5          ; There are three possible stack configurations.  The users system
6          ; stack is stored in the each job's context region.  This stack
7          ; is bounded by JSTK (top of stack) and JSTKND (bottom of stack).
8          ; The interrupt stack is used during interrupt processing and is
9          ; allocated over TSINIT code.  The address boundary of this stack
10         ; is stored in INTSTK (top of stack) and INTSND (bottom of stack)
11         ; located in TSEXEC.  The system stack is used during scheduling
12         ; and is located between SS (top of stack) and SSEND (bottom of stack).
13         ;
14 002072  STACK:  PRINT  #CRLF          ;Display CR/LF
15 002102  013705  0000000  MOV    DIESP,R5      ;Get the stack pointer
16 002106  010504          MOV    R5,R4        ;Copy the base of the stack
17 002110  020527  0000000  5#:   CMP    R5,#SS   ;Below the top of the system stack?
18 002114  101013          BHI    10#          ;No, SP not in system stack
19         ;
20         ; Stack pointer is in the system stack area.
21         ;
22 002116  162704  0000000          SUB    #SS,R4      ;Calculate number of bytes to dump
23 002122  001465          BEQ    50#         ;No stack space used
24 002124  005404          NEG    R4          ;Convert to positive byte count
25 002126          PRINT  #STKSS      ;Display "System Stack Contents"
26 002136  004737  002640'  CALL  OSTACK      ;Output the stack contents
27 002142  000207          RETURN
28 002144  020537  0000000  10#:  CMP    R5,INTSTK ;Below the top of the interupt stack?
29 002150  101013          BHI    20#         ;No, SP not in interrupt stack
30         ;
31         ; Stack pointer is in the interrupt stack area.
32         ;
33 002152  163704  0000000          SUB    INTSTK,R4  ;Calculate number of bytes to dump
34 002156  001447          BEQ    50#         ;No stack space used
35 002160  005404          NEG    R4          ;Convert to positive byte count
36 002162          PRINT  #STKINT     ;Display "Interrupt Stack Contents"
37 002172  004737  002640'  CALL  OSTACK      ;Output the stack contents
38 002176  000207          RETURN
39 002200  020527  0000000  20#:  CMP    R5,#JSTKND ;Above the end of the context stack?
40 002204  103416          BLO    30#         ;Invalid stack pointer
41 002206  020527  0000000  CMP    R5,#JSTK   ;Below the top of the context stack?
42 002212  101013          BHI    30#         ;No, SP not in job context stack
43         ;
44         ; Stack pointer is in the job's context stack.
45         ;
46 002214  162704  0000000  21#:  SUB    #JSTK,R4   ;Calculate number of bytes to dump
47 002220  001426          BEQ    50#         ;No stack space used
48 002222  005404          NEG    R4          ;Convert to positive byte count
49 002224          PRINT  #STKEMT     ;Display "EMT Stack Contents"
50 002234  004737  002640'  CALL  OSTACK      ;Output the stack contents
51 002240  000207          RETURN
52         ;
53         ; Stack pointer is invalid.  Output 40 words where SP is pointing.
54         ;
55 002242  30#:  PRINT  #INVSTK     ;Display "Invalid stack"
56 002252  020527  0000000  CMP    R5,#VPAR5  ;Check for EMT stack overflow
57 002256  103403          BLO    40#         ;Br if below 120000, unknown stack

```

STACK -- Display contents of stack

58	002260	012705	0000000		MOV	#VPAR6,R5	;Adjust stack pointer to EMT stack
59	002264	000753			BR	21\$	;Dump job context stack
60	002266	012704	000310	40\$:	MOV	#100,*2,R4	;Dump 100. words from stack pointer
61	002272	004737	002640'		CALL	DSTACK	;Output the stack contents
62	002276	000207		50\$:	RETURN		

MISC -- Display misc system values

```

1          .SBTTL  MISC  -- Display misc system values
2          ;-----
3          ; MISC is called to display the miscellaneous system values.
4          ;
5 002300   MISC:
6 002300           PRINT  #CRLF          ;Display CR/LF
7 002310   105737 0000000  TSTB  INTLVL      ;Check interrupt level
8 002314   002404           BLT    10$      ;Br if not executing interrupt
9 002316           PRINT  #INTLEV       ;Display "Interrupt level"
10 002326  105737 0000000  10$: TSTB  FRKPRI   ;Check fork priority
11 002332   001404           BEQ    20$      ;Br if not executing fork
12 002334           PRINT  #FRKLEV      ;Display "Fork level"
13 002344  023727 0000000 123456 20$: CMP    SSEND,#123456 ;Check end of system stack
14 002352   001404           BEQ    30$      ;Br if stack end is valid
15 002354           PRINT  #STKOVR     ;Display "System stack overflow"
16 002364   005002  30$:  CLR    R2         ;Clear offset pointer
17 002366   012705 000164'  MOV    #VMISC,R5   ;Point to miscellaneous output list
18 002372   004737 002504'  CALL  OLIST      ;Output the list of values
19 002376   000207           RETURN

```

JOB -- Display info about current job

```

1          .SBTTL  JOB  -- Display info about current job
2          ;-----
3          ;  JOB is called to display information concerning the executing job.
4          ;
5 002400  JOB:    PRINT  #CRLF          ;Display CR/LF
6 002410          PRINT  #CURJOB       ;Disp "Job executing at time of dump"
7 002420          PRINT  #EQSTR        ;Display " = "
8 002430  113701 0000000  MOVB  CORUSR,R1    ;Get the current operating job index
9 002434  006201          ASR    R1      ;Divide by two for the job number
10 002436  012703 000002  MOV   #$DEC,R3    ;Force decimal display
11 002442  004737 003626' CALL  OUTVAL     ;Output the job number
12 002446          PRINT  #CRLF          ;Display CR/LF
13 002456  000207          RETURN

```

LINE -- Display line definition tables

```
1 .SBTTL LINE -- Display line definition tables
2 ;-----
3 ; LINE is called to display the generated line definition tables.
4 ;
5 002460 012705 000622' LINE: MOV #JLIN,R5 ;Point to job line table
6 002464 004737 002714' CALL OTABLE ;Output line definition values
7 002470 000207 RETURN
```

DEVICE -- Display device definition tables

```
1 .SBTTL DEVICE -- Display device definition tables
2 ;-----
3 ; DEVICE is called to display the generated device definition tables.
4 ;
5 002472 012705 001410' DEVICE: MOV #DHAN,R5 ;Point to device table
6 002476 004737 002714' CALL OTABLE ;Output device table values
7 002502 000207 RETURN
```

OLIST -- Output a list of information

```

1          .SBTTL  OLIST  -- Output a list of information
2          ;-----
3          ;  Output a list of information.
4          ;
5 002504  010546  OLIST:  MOV      R5,-(SP)      ;Save registers
6 002506  010446          MOV      R4,-(SP)
7 002510  010346          MOV      R3,-(SP)
8 002512  010246          MOV      R2,-(SP)
9 002514  010146          MOV      R1,-(SP)
10         ;
11         ;  Display the list header
12         ;
13 002516  012501          MOV      (R5)+,R1      ;Point to list header
14 002520  001437          BEQ      100$          ;Br if not header
15 002522          PRINT          ;Display the list header
16 002526  000434          BR       100$          ;Output entire display list
17         ;
18         ;  Display identifier
19         ;
20 002530  10$:  PRINT      (R5)          ;Print identifier
21 002536  30$:  PRINT      #EQSTR        ;Point to string " = "
22 002546  016503  000002      MOV      2(R5),R3      ;Get field attributes
23 002552  016504  000004      MOV      4(R5),R4      ;Get the field address
24 002556  060204          ADD      R2,R4          ;Add field bias
25 002560  032703  000001      BIT      ##BYTE,R3    ;Word or byte storage element?
26 002564  001404          BEQ      31$          ;Br if word element
27 002566  111401          MOVB   @R4,R1          ;Get numeric value
28 002570  042701  177400      BIC      #^C377,R1     ;Clear sign extend
29 002574  000401          BR       32$          ;Continue
30 002576  011401  31$:  MOV      @R4,R1          ;Get the numeric value
31 002600  004737  003626'  32$:  CALL      OUTVAL        ;Output the numeric value
32 002604          PRINT      #CRLF          ;Point to carriage return / line feed
33 002614  062705  000006  50$:  ADD      #6,R5          ;Increment to next list entry
34         ;
35         ;  Continue output of list elements.
36         ;
37 002620  005715  100$:  TST      (R5)          ;Point to the next list entry
38 002622  001342          BNE     10$          ;Br if list continues
39         ;
40         ;  Finished
41         ;
42 002624  012601          MOV      (SP)+,R1      ;Restore registers
43 002626  012602          MOV      (SP)+,R2
44 002630  012603          MOV      (SP)+,R3
45 002632  012604          MOV      (SP)+,R4
46 002634  012605          MOV      (SP)+,R5
47 002636  000207          RETURN          ;Finished

```

OSTACK -- Display stack contents

1			.SBTTL OSTACK -- Display stack contents
2			-----
3			; Output the stack contents.
4			;
5	002640		OSTACK:
6	002640	006204	ASR R4 ; Convert byte count to word count
7	002642		PRINT #STKCNT ; Display header
8	002652	005003	CLR R3 ; Set for octal display
9	002654	010501	10\$: MOV R5,R1 ; Copy the stack address
10	002656	004737 003626'	CALL OUTVAL ; Display stack address
11	002662		PRINT #STKDLM ; Display delimiter
12	002672	012501	MOV (R5)+,R1 ; Get the stack contents
13	002674	004737 003626'	CALL OUTVAL ; Display stack contents
14	002700		PRINT #CRLF ; Display CR/LF
15	002710	077417	SOB R4,10\$ ; Continue until done
16	002712	000207	RETURN

OTABLE -- Output tables of information

```
1                                     .SBTTL  OTABLE -- Output tables of information
2                                     ;-----
3                                     ; Output tables of information.
4                                     ;
5 002714                               OTABLE:
6 002714 004737 002752'                CALL  TINIT          ;Initialize for table output
7 002720 004737 003006'                1$:  CALL  TCLR          ;Clear output tables
8 002724 004737 003156'                CALL  TACCR         ;Accrue table information
9 002730 004737 003432'                CALL  TOUT         ;Display table values
10 002734 005715                        TST    (R5)         ;Check for more table information
11 002736 001370                        BNE    1$           ;Br if more to display
12 002740                                PRINT #CRLF       ;Display CR/LF
13 002750 000207                        RETURN
```

OTABLE -- Output tables of information

```

1          ;
2          ; Initialize for display start or continuation of table format.
3          ;
4 002752   TINIT:  PRINT  #CRLF          ;Display CR/LF
5 002762           PRINT  (R5)+        ;Display the table title
6 002770   005037  000052'           CLR    COL          ;Clear column counter
7 002774           PRINT  #CRLF          ;Display CR/LF
8 003004   000207           RETURN
9          ;
10         ; Clear table entries for continuation of table format.
11         ;
12 003006   012702  000014   TCLR:  MOV    #NCOL,R2      ;Initilize table storage information
13 003012   006302           ASL    R2              ;Convert to word index
14 003014   005062  000054'   10$:  CLR    TSCOL(R2)      ;Clear starting column number
15 003020   005062  000106'           CLR    TECOL(R2)      ;Clear ending column number
16 003024   005062  000140'           CLR    TSINX(R2)     ;Clear starting index number
17 003030   005062  000172'           CLR    TEINX(R2)     ;Clear ending index number
18 003034   005062  000224'           CLR    TATT(R2)     ;Clear attribute flags
19 003040   005062  000256'           CLR    TADR(R2)     ;Clear address location
20 003044   162702  000002           SUB    #2,R2        ;Subtract from table index
21 003050   003361           BGT    10$          ;Continue for all entries
22 003052   002427           BLT    20$          ;All entries are clear
23 003054   005737  000052'           TST    COL          ;Check column counter
24 003060   001755           BEQ    10$          ;Br if first table initialization call
25         ;
26         ; Table continuation display.
27         ;
28 003062           PRINT  #CRLF          ;Print CR/LF
29 003072           PRINT  #CRLF          ;Print CR/LF
30 003102           PRINT  TADR(R2)       ;Print columnar heading
31 003112   016201  000106'           MOV    TECOL(R2),R1 ;Find ending column information
32 003116   004737  004144'           CALL   BLKPAD        ;Blank fill
33 003122           PRINT  #SPSTR        ;Print blanks for spacing
34 003132   000207   20$:  RETURN

```

OTABLE -- Output tables of information

```

1
2 ; Accrue information concerning table display while printing column headings.
3 ;
4 003134 ; NXTCOL:
5 003134 016562 000004 000256' MOV 4(R5),TADR(R2) ; Store print string address
6 003142 062705 000012 ADD #10,R5 ; Offset to the next column entry
7 003146 PRINT #SPSTR ; Print blanks for spacing
8 003156 062702 000002 TACCR: ADD #2,R2 ; Increment the field index
9 003162 020227 000014 CMP R2,#NXC:OL ; End of columns allowed?
10 003166 002120 BGE 50$ ; Br is no more column storage
11 003170 005715 TST (R5) ; See if a column entry exists
12 003172 001516 BEQ 50$ ; End of table found
13 003174 013762 000052' 000054' MOV COL,TSCOL(R2) ; Point to starting field position
14 003202 PRINT (R5) ; Display the column heading
15 003210 013762 000052' 000106' MOV COL,TECOL(R2) ; Set ending column position
16 003216 016562 000002 000224' MOV 2(R5),TATT(R2) ; Store field attributes
17 003224 016500 000006 MOV 6(R5),R0 ; Get the starting index number
18 003230 001002 BNE 1$ ; Br if start is specified
19 003232 013700 000140' MOV TSINX,R0 ; Set as starting table number
20 003236 010062 000140' 1$: MOV R0,TSINX(R2) ; Store starting index number
21 003242 016500 000010 MOV 10(R5),R0 ; Get the ending index number
22 003246 001002 BNE 2$ ; Br if end is specified
23 003250 013700 000172' MOV TEINX,R0 ; Set as ending table number
24 003254 010062 000172' 2$: MOV R0,TEINX(R2) ; Store ending index number
25 003260 012700 000006 MOV #6,R0 ; Assume 6 digit (word element)
26 003264 032762 000001 000224' BIT ##BYTE,TATT(R2) ; Check field attributes
27 003272 001402 BEQ 10$ ; Br if word element
28 003274 012700 000003 MOV #3,R0 ; Assume 3 digit (byte element)
29 003300 066200 000054' 10$: ADD TSCOL(R2),R0 ; Add starting position
30 003304 026200 000106' CMP TECOL(R2),R0 ; Determine if ending position
31 003310 001427 BEQ 30$ ; Field is exactly enough
32 003312 101013 BHI 20$ ; Br if field too long
33 003314 010001 MOV R0,R1 ; Copy ending column address
34 003316 166201 000106' SUB TECOL(R2),R1 ; Find difference
35 003322 010062 000106' MOV R0,TECOL(R2) ; Force ending column longer
36 003326 112700 000040 MOVB #' ,R0 ; Store blank
37 003332 11$: TTYOUT ; Pad output with blanks
38 003336 077103 SOB R1,11$ ; Continue until padded
39 003340 000413 BR 30$ ; Go on to the next entry
40 003342 166200 000106' 20$: SUB TECOL(R2),R0 ; Subtract the ending column
41 003346 032762 040000 000224' BIT ##RJUST,TATT(R2) ; Check for right justify
42 003354 001403 BEQ 21$ ; Br if left justify
43 003356 160062 000054' SUB R0,TSCOL(R2) ; Adjust starting column position
44 003362 000402 BR 30$
45 003364 060062 000106' 21$: ADD R0,TECOL(R2) ; Adjust ending column position
46 003370 026227 000106' 000120 30$: CMP TECOL(R2),#30. ; Check for line overflow
47 003376 103656 BLO NXTCOL ; Continue collecting columns
48 003400 005062 000054' CLR TSCOL(R2) ; Clear partial entry
49 003404 005062 000106' CLR TECOL(R2)
50 003410 005062 000140' CLR TSINX(R2)
51 003414 005062 000172' CLR TEINX(R2)
52 003420 005062 000224' CLR TATT(R2)
53 003424 005062 000256' CLR TADR(R2)
54 003430 000207 50$: RETURN

```

OTABLE -- Output tables of information

```

1      ;
2      ;   Display index value and data.
3      ;
4      ;
5 003432 010446   TOUT:   MOV     R4,-(SP)      ;Save some registers
6 003434 010546           MOV     R5,-(SP)
7      ;
8      ;   Display index offset into table.
9      ;
10 003436 013705 000140'           MOV     TSINX,R5      ;Find the starting index
11 003442           1$:   PRINT  #CRLF      ;Display CR/LF
12 003452 010501           MOV     R5,R1        ;Copy index number
13 003454 005002           CLR     R2            ;Clear column index
14 003456 016203 000224'           MOV     TATT(R2),R3   ;Numeric attribute
15 003462 004737 003626'           CALL   OUTVAL        ;Display index value
16 003466 000430           BR     40$          ;Enter column output stream
17      ;
18      ;   Determine if the value should be displayed.
19      ;
20 003470 020562 000140'   10$:   CMP     R5,TSINX(R2)  ;Is the current index < starting
21 003474 002420           BLT    40$          ;Yes, skip entry
22 003476 020562 000172'           CMP     R5,TEINX(R2) ;Is the current index > ending
23 003502 003022           BGT    40$          ;Yes, skip entry
24      ;
25      ;   Output rows of information.
26      ;
27 003504 004737 004144'           CALL   BLKPAD        ;Pad with spaces
28 003510 016203 000224'           MOV     TATT(R2),R3   ;Get the attribute
29 003514 016201 000256'           MOV     TADR(R2),R1   ;Check for existing entry
30 003520 060501           ADD     R5,R1        ;Add the index offset (byte)
31 003522 032703 000001           BIT     ##BYTE,R3    ;Check for byte storage
32 003526 001404           BEQ    20$          ;Br if word storage
33 003530 111101           MOVB   (R1),R1       ;Get the stored byte value
34 003532 042701 177400           BIC    #^C377,R1    ;Kill sign extend bits
35 003536 000402           BR     30$          ;Continue
36 003540 060501           20$:   ADD     R5,R1        ;Add the index offset (word)
37 003542 011101           MOV     (R1),R1      ;Get the stored word value
38 003544 004737 003626'   30$:   CALL   OUTVAL        ;Output the value
39 003550 062702 000002           40$:   ADD     #2,R2        ;Increment the index
40 003554 016201 000054'           MOV     TSCOL(R2),R1 ;Get the starting column
41 003560 001343           BNE    10$          ;End of table entries
42 003562 005205           INC     R5            ;Increment the index offset
43 003564 020537 000172'           CMP     R5,TEINX     ;Check for ending index
44 003570 101724           BLOS   1$            ;Br if more tables to output
45 003572 012605           MOV     (SP)+,R5     ;Restore registers
46 003574 012604           MOV     (SP)+,R4
47 003576 000207           RETURN

```

OUTSTR -- Output an asciz string

```

1          .SBTTL  OUTSTR -- Output an asciz string
2          ;-----
3          ;  OUTSTR - Output an asciz string.
4          ;
5          ;  Arguments --
6          ;    R1      =      String address
7          ;
8
9 003600   OUTSTR:
10 003600   112100  1$:      MOVB      (R1)+, R0      ;Get next output byte
11 003602   001404          BEQ      10$      ;Zero, output CR / LF
12 003604   100407          BMI      20$      ;Exit on end of output string
13 003606          TTYOUT      ;Output the character
14 003612   000772          BR      1$      ;Continue until end of string
15 003614   10$:      PRINT      #CRLF      ;Display carriage return / line feed
16 003624   000207  20$:      RETURN

```

OUTVAL -- Convert and print a value

```

1          .SBTTL  OUTVAL -- Convert and print a value
2          ;-----
3          ;  OUTVAL -- Convert and output a value.
4          ;
5          ;  Arguments --
6          ;      R1      = Numeric value
7          ;      R3      = Display string attributes
8          ;
9 003626 010546 OUTVAL: MOV      R5,-(SP)      ;Save registers
10 003630 010446      MOV      R4,-(SP)
11 003632 010346      MOV      R3,-(SP)
12 003634 010246      MOV      R2,-(SP)
13 003636 032703 000004 BIT      ##RAD50,R3      ;Check for RAD50 value
14 003642 001403      BEQ      1$          ;Br if not RAD50
15 003644 004737 004022' CALL     OUTRAD      ;Output RAD50
16 003650 000420      BR       30$          ;Continue
17 003652 012704 000010 1$:      MOV      #10,R4      ;Assume octal base
18 003656 032703 000002 BIT      ##DEC,R3      ;Check for decimal base
19 003662 001402      BEQ      10$         ;Br if numeric display in octal
20 003664 012704 000012      MOV      #10.,R4     ;Set for decimal base
21 003670 012702 000006 10$:     MOV      #6,R2        ;Assume word value
22 003674 032703 000001 BIT      ##BYTE,R3     ;Check for byte storage
23 003700 001402      BEQ      20$         ;Br if storage is word value
24 003702 012702 000003      MOV      #3,R2        ;Otherwise byte value
25 003706 004737 003724' 20$:     CALL     CONVAL      ;Convert numeric value
26 003712 012602 30$:     MOV      (SP)+,R2     ;Restore registers
27 003714 012603      MOV      (SP)+,R3
28 003716 012604      MOV      (SP)+,R4
29 003720 012605      MOV      (SP)+,R5
30 003722 000207      RETURN
31          ;
32 003724      CONVAL:
33 003724 005302      DEC      R2          ;Say one more digit converted
34 003726 002411      BLT      2$          ;Br if all display digits converted
35 003730 005000      CLR      R0          ;Clear high order result
36 003732 071004      DIV      R4,R0        ;Divide number by numeric base
37 003734 062701 000060      ADD      #60,R1      ;Convert remainder to ascii digit
38 003740 010146      MOV      R1,-(SP)     ;Save ascii digit on stack
39 003742 010001      MOV      R0,R1        ;Set result register
40 003744 001406      BEQ      3$          ;Br if no more digits to convert
41 003746 004737 003724'      CALL     CONVAL      ;Call conversion for next digit
42 003752 112600 2$:      MOVB     (SP)+,R0     ;Get the next digit to display
43 003754      TTYOUT      ;Output the ascii character
44 003760 000207      RETURN
45 003762 005702 3$:      TST      R2          ;Check field output size
46 003764 001772      BEQ      2$          ;No pad necessary
47 003766 032703 040000      BIT      ##RJUST,R3   ;Check for right justify
48 003772 001767      BEQ      2$          ;Br if not right justify
49 003774 112700 000040      MOVB     #' ',R0      ;Default pad to blank
50 004000 032703 020000      BIT      ##ZRPAD,R3   ;Check for zero pad
51 004004 001402      BEQ      4$          ;Br if blank pad chosen
52 004006 112700 000060      MOVB     #'0',R0     ;Change to zero pad
53 004012 4$:      TTYOUT      ;Output the character
54 004016 077203      SOB      R2,4$      ;Continue until all output
55 004020 000754      BR       2$          ;Display remaining characters

```

```

1          .SBTTL  OUTRAD -- Convert and print a RAD50 value
2          ;-----
3          ;  OUTRAD - Convert and output rad50 data.
4          ;      R1      = Numeric rad50 value
5          ;
6 004022  010146  OUTRAD: MOV     R1,-(SP)
7 004024  005000          CLR     R0          ;Clear high order
8 004026  071027  003100      DIV    #50*50,R0      ;Divide for 1st byte
9 004032          TTYOUT  R50CHR(R0)  ;Display rad50 character
10 004042  005000          CLR     R0          ;Clear high order
11 004044  071027  000050      DIV    #50,R0        ;Divide for 2nd byte
12 004050          TTYOUT  R50CHR(R0)  ;Display rad50 character
13 004060          TTYOUT  R50CHR(R1)  ;Display rad50 character
14 004070  012601          MOV     (SP)+,R1
15 004072  000207          RETURN
16          ;
17 004074      040      101      102  R50CHR: .ASCII / ABCDEF0HIJKLMNOPQRSTUVWXYZ#. 0123456789/
18          004077      103      104      105
19          004102      106      107      110
20          004105      111      112      113
21          004110      114      115      116
22          004113      117      120      121
23          004116      122      123      124
24          004121      125      126      127
25          004124      130      131      132
26          004127      044      056      040
27          004132      060      061      062
28          004135      063      064      065
29          004140      066      067      070
30          004143      071
18          .EVEN
19
20          .SBTTL  BLKPAD -- Blank pad output stream
21          ;-----
22          ;  BLKPAD - Blank pad ouput stream.
23          ;      R1      = ending column number
24          ;
25 004144  112700  000040  BLKPAD: MOVB   #' ,R0          ;Initialize ouput to space
26 004150  163701  000052'      SUB     COL,R1          ;Figure number of blanks
27 004154  003403          BLE     20$           ;Br if already past column
28 004156          10$:   TTYOUT          ;Display space
29 004162  077103          SOB     R1,10$        ;Continue
30 004164  000207          20$:   RETURN

```

OUTCHR -- Output an ascii character

```

1          .SBTTL  OUTCHR -- Output an ascii character
2          ;-----
3          ;  OUTCHR - Output an ascii character to a serial or parallel port.
4          ;      RO      = character to output
5          ;
6 004166 005737 000050'  OUTCHR: TST      RCSR      ; Check for receiver buffers
7 004172 001415          BEQ      OPRINT   ; No receiver CSR, transmit character
8 004174          OTERM:
9 004174 105777 173650 10$:  TSTB     @RCSR    ; Check input port
10 004200 100005          BPL      20$     ; Br if no input pending
11 004202 117737 173640 000340'  MOVB     @RBUF, XFLG  ; Input the character
12 004210 142737 177600 000340'  BICB     #^C177, XFLG ; Strip eighth bit
13 004216 123727 000340' 000021 20$:  CMPB     XFLG, #CTRLQ ; Check for XDN
14 004224 001363          BNE      10$     ; Br if not XDN state active
15 004226          OPRINT:
16 004226 105777 173612 30$:  TSTB     @TCSR    ; Check transmit done flag
17 004232 100375          BPL      30$     ; Wait for transmit done
18 004234 110077 173602          MOVB     RO, @TBUF   ; Output to transmitter
19 004240 120027 000015          CMPB     RO, #CK     ; Is character a carriage return?
20 004244 001003          BNE      40$     ; Br if not carriage return
21 004246 005037 000052'  CLR      COL      ; Clear column counter
22 004252 000405          BR       50$     ; Finished
23 004254 120027 000040 40$:  CMPB     RO, #40    ; Is character printable?
24 004260 103402          BLO      50$     ; Br if not printable
25 004262 005237 000052'  INC      COL      ; Increment column position
26 004266 000207 50$:  RETURN   ; Finished
27
28          .END

```

Errors detected: 0

\*\*\* Assembler statistics

Work file reads: 0  
 Work file writes: 0  
 Size of work file: 9623 Words ( 38 Pages)  
 Size of core pool: 17920 Words ( 70 Pages)  
 Operating system: RT-11

Elapsed time: 00:00:58.48  
 DK: TSDUMP, LP: TSDUMP=DK: TSDUMP, MAC/C/N: SYM



DIEPC	2-11	7-13	7-13	
DIESP	2-11	7-12	7-12	20-15
DIV32	14-33	14-37	15-8#	
DMPHND	2-11	16-18		
DMPOVL	2-11	16-12		
DMPTRP	13-32	13-76#		
DMPTXT	2-11	16-6		
DODUMP	2-4	13-27#		
DOSCHD	9-12	9-12		
DVFLAG	11-11	11-11		
DVSTAT	11-12	11-12		
EQSTR	12-72#	19-17	22-7	25-21
ERROR	13-63	16-5#		
EXCJOB	9-13	9-13		
FF	12-74#	13-61		
FREFRK	9-16	9-16		
FREIOQ	9-14	9-14		
FRKCQE	9-15	9-15		
FRKLEV	12-84#	21-12		
FRKPRI	9-17	9-17	21-10	
HANENT	11-7	11-7		
HANIOC	11-10	11-10		
HANPAR	11-8	11-8		
HANSIZ	11-9	11-9		
INBSY	9-18	9-18		
INTLEV	12-83#	21-9		
INTLVL	9-19	9-19	21-7	
INTPRI	9-20	9-20		
INTSND	2-9			
INTSTK	2-9	20-28	20-33	
INVSTK	12-82#	20-55		
IQABFL	9-21	9-21		
JLIN	10-5#	23-5		
JOB	13-64	22-5#		
JSTK	2-9	20-41	20-46	
JSTKND	2-9	20-39		
KPAR6	2-13	13-51		
LACTIV	10-24	10-24		
LAFSIZ	10-34	10-34		
LBASE	10-18	10-18		
LRSPRI	10-30	10-30		
LCDTYP	10-27	10-27		
LCLUNT	10-31	10-31		
LCMPL	10-36	10-36		
LEMTPC	10-39	10-39		
LF	12-8#	12-75		
LFWLIM	10-35	10-35		
LINCNT	10-23	10-23		
LINE	13-69	23-5#		
LIOCNT	10-22	10-22		
LIOHLD	10-33	10-33		
LJSW	10-38	10-38		
LMEMIN	10-32	10-32		
LNBLKS	10-20	10-20		
LNPRIM	10-28	10-28		
LNSBLK	10-21	10-21		







