

Table of contents

8-	1	Output list for registers
9-	1	Output list for TSX-Plus addresses
10-	1	Output list for miscellaneous fields
11-	1	Output list for line status tables
12-	1	Output list for device handler status
14-	1	Output list for job context information
15-	1	Data areas
16-	1	DODUMP -- Dump entry point
17-	1	DATTIM -- Display date and time information
18-	1	DIV32 -- Divide 16-bit into 32-bit
19-	1	ERROR -- Display error status information
20-	1	REG -- Display contents of registers
21-	1	MODADR -- Display TSX.SAV module addresses
22-	1	OVRADR -- TSX-Plus overlay region
23-	1	STACK -- Display contents of stack
24-	1	MISC -- Display misc system values
25-	1	JOB -- Display info about current job
26-	1	LINE -- Display line definition tables
27-	1	DEVICE -- Display device definition tables
28-	1	FORK -- Display fork block information
29-	1	JCBINF -- Display job context information
30-	1	OLIST -- Output a list of information
31-	1	OSTACK -- Display stack contents
32-	1	OTABLE -- Output tables of information
36-	1	OUTSTR -- Output an asciz string
37-	1	OUTVAL -- Convert and print a value
38-	1	OUTRAD -- Convert and print a RAD50 value
38-	20	BLKPAD -- Blank pad output stream
39-	1	OUTCHR -- Output an ascii character

```
1          .TITLE  TSDUMP  -- TSX-Plus crash dump
2          .ENABL  LC
3          .DSABL  GBL
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      ; -----
7      ; GLOBAL References.
8      ;
9      .GLOBL SS, SSEND, INTSTK, INTSND, JSTK, JSTKND
10     .GLOBL VDMTCR, PSW
11     .GLOBL DMPTXT, DMPOVL, DMPHND, DIEARG, DIEPC, DIESP
12     .GLOBL TSGEN, TSTIO, 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
20     .GLOBL $INCOR                ; LSW in memory flag
21     .GLOBL FQ$LNK                ; Link word to next fork block
22     .GLOBL LCXPAR                ; PAR relocation base of job context
23     .GLOBL RT11EX                ; RT-11 function bias

```

```

1      ; -----
2      ;  MACRO Definitions.
3      ; -----
4      ;  DISABLE macro used to disable all interrupts.
5      ; -----
6      ;
7      ;      .MACRO  DISABL          ;DISABLE INTERRUPTS
8      ;      BIS      #340, @#PSW
9      ;      .ENDM   DISABL
10     ; -----
11     ;  STKDEF macro used to define a stack where:
12     ;
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     ;  PRINT macro used to define a print string where
24     ;      stradr    = address of asciz string to print
25     ; -----
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     ;  TTYOUT macro used to output a single character where
36     ;      char      = character to output
37     ; -----
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      ; PVAL macro used to define a single print field where:
3      ;     NAME      = ascii string title
4      ;     LABEL     = address label containing print value
5      ;     ATTR      = field attributes
6      ;     SINX      = starting table index
7      ;     EINX      = ending table index
8      ;     Format attributes:
9      ;     $RJUST = (bit 14) right justify (default is left)
10     ;     $ZRPAD = (bit 13) zero pad (default is blank)
11     ;     Storage attributes:
12     ;     $BYTE  = (bit 0) storage requirement (default is word)
13     ;     $DEC   = (bit 2) decimal representation (default is octal)
14     ;     $RAD50 = (bit 3) rad50 representation
15     ;
16
17     ; Definition of field attribute bit definitions.
18
19     040000      $RJUST = 40000      ; Right justify
20     020000      $ZRPAD = 20000      ; Zero pad
21
22     ; Storage attributes.
23     000001      $BYTE  = 1          ; Byte storage cell
24     000002      $DEC   = 2          ; Decimal numeric representation
25     000004      $RAD50 = 4          ; Rad50 representation
26
27     .MACRO PVAL name, label, attr, sindx, eindx
28     .PSECT STRING ; Define STRING program section
29     STRADD = ; Point to the starting address
30     .ASCII /name/ ; Store the ascii label
31     .BYTE 200 ; Terminate as asciz string
32     .PSECT DUMP ; Define DUMP program section
33     .WORD STRADD ; Store the string address
34     .WORD attr ; Store field attributes
35     .GLOBL label ; Declare variable as global
36     .WORD label ; Store location of value field
37     .IF NE, tenabl ; Has table definition been enabled?
38     .WORD sindx ; Store starting table index
39     .WORD eindx ; Store ending table index
40     .ENDC ; NE, tenabl
41     .PSECT TSDUMP ; Restore TSDUMP program section
42     .ENDM PVAL

```

```

1      ; -----
2      ; PVAL and PRVAL are identical with the exception that PVAL
3      ; declares the entity as GLOBL and PRVAL does not.
4
5      .MACRO PRVAL name, label, attr, sindx, eindx
6      .PSECT STRING ; Define STRING program section
7      STRADD = ; Point to the starting address
8      .ASCII /name/ ; Store the ascii label
9      .BYTE 200 ; Terminate as asciz string
10     .PSECT DUMP ; Define DUMP program section
11     .WORD STRADD ; Store the string address
12     .WORD attr ; Store field attributes
13     .WORD label ; Store location of value field
14     .IF NE, tenabl ; Has table definition been enabled?
15     .WORD sindx ; Store starting table index
16     .WORD eindx ; Store ending table index
17     .ENDC ; NE, tenabl
18     .PSECT TSDUMP ; Restore TSDUMP program section
19     .ENDM PRVAL

```

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.  
      ;  
      TBEG    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
```

Output list for TSX-Plus addresses

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.
;

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

```

Output list for line status tables

```

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

```

Output list for device handler status

```

1          .SBTTL  Output list for device handler status
2          ;-----
3          ;  Generate table for output of device handler status.
4          ;
5 000002    TBEG    DHAN,<Device Handler Tables>,<Offset>,<#BYTE!$DEC>,0,MAXDEV-1
6 000002    PVAL    <PNAME>,PNAME,<#RAD50>          ;Program name
7 000002    PVAL    <HANENT>,HANENT,<#RJUST>          ;Handler entry point
8 000002    PVAL    <HANPAR>,HANPAR,<#RJUST>          ;Handler PAR
9 000002    PVAL    <HANSIZ>,HANSIZ,<#RJUST>          ;Handler size
10 000002   PVAL    <HANIOC>,HANIOC,<#RJUST>          ;Handler I/O count
11 000002   PVAL    <DVFLAG>,DVFLAG,<#RJUST!$ZRPAD> ;Device handler flags
12 000002   PVAL    <DVSTAT>,DVSTAT,<#RJUST!$ZRPAD> ;Device status flags
13 000002   PVAL    <DEVSIZ>,DEVSIZ,<#RJUST!$ZRPAD> ;Device size (256 word blks)
14 000002   TEND

```

Output list for device handler status

```
1  
2  
3  
4 000002      TBEG      FRKREQ, <Current FORK request>  
5 001550      PVAL      <Address>, FQ$RTN      ; Address of FORK routine  
6 000002      PVAL      <Saved R5>, FQ$R5      ; Saved value of R5  
7 000002      PVAL      <Saved R4>, FQ$R4      ; Saved value of R4  
8 000002      PVAL      <Saved R3>, FQ$R3      ; Saved value of R3  
9 000002      PVAL      <Saved R2>, FQ$R2      ; Saved value of R2  
10 000002     PVAL      <Saved R1>, FQ$R1      ; Saved value of R1  
11 000002     PVAL      <User's FORK block add>, FQ$UFB ; User specified FORK block add  
12 000002     PVAL      <Saved KPAR5>, FQ$PA5    ; Saved value of KPAR5  
13 000002     PVAL      <Saved KPAR6>, FQ$PA6    ; Saved value of KPAR6  
14 000002     PVAL      <FORK Priority>, FQ$PRI, <$BYTE> ; FORK priority  
15 000002      TEND
```

Output list for job context information

```

1          .SBTTL  Output list for job context information
2          ;-----
3          ;  Generate list for output of job context information.
4          ;
5
6 000002      TBEG      VJCB, <Job context information>; Beginning of print list
7 001650      PVAL      <URO>, URO                      ; User r0 return information
8 000002      PVAL      <CUREMT>, CUREMT                  ; Current emt
9 000002      PVAL      <EMTBLK+0 >, EMTBLK, <$BYTE>      ; EMT channel number
10 000002     PRVAL     <      +1 >, EMTFUN, <$BYTE>      ; EMT function code
11 000002     PRVAL     <      +2 >, EMTBLK+2              ; EMT argument block
12 000002     PRVAL     <      +4 >, EMTBLK+4              ; EMT argument block
13 000002     PRVAL     <      +6 >, EMTBLK+6              ; EMT argument block
14 000002     PRVAL     <     +10>, EMTBLK+10             ; EMT argument block
15 000002     PRVAL     <     +12>, EMTBLK+12             ; EMT argument block
16 000002     PRVAL     <     +14>, EMTBLK+14             ; EMT argument block
17 000002     PRVAL     <     +16>, EMTBLK+16             ; EMT argument block
18 000002     PVAL      <CHNADR>, CHNADR                  ; Channel address
19 000002      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 000016 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         000054 000000 SAVKP6: .WORD  0          ; Saved KPAR6 for remapping
48         000056 000000 EMTFUN: .WORD  0          ; EMT function code
49         ;
50         ; Format tables for columnar output.
51         ;
52         000060 TSCOL:  .BLKW  NCOL+1          ; Starting column on table output
53         000112 TECOL:  .BLKW  NCOL+1          ; Ending column on table output
54         000144 TSINX:  .BLKW  NCOL+1          ; Starting index for column
55         000176 TEINX:  .BLKW  NCOL+1          ; Ending index for column
56         000230 TATT:   .BLKW  NCOL+1          ; Attributes of table
57         000262 TADR:   .BLKW  NCOL+1          ; Base address of table

```

Data areas

```

58      ;
59      ; Data.
60      ;
61 000314 037266 023112 050572 MONTH: .RAD50 /JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC/
      000322 004322 050601 040726
      000330 040724 004617 073630
      000336 057114 054756 014713
62      ;
63      ; Byte bounded storage.
64      ;
65 000344      021      XFLG: .BYTE CTRLQ ;XON/XOFF flag
66      ;
67      ; Output strings.
68      ;
69      .NLIST BEX
70 000345      101      162      147 TXARG: .ASCII /Arg. value = /<200>
71 000363      123      145      147 TXSEG: .ASCII /Seg. value = /<200>
72 000401      117      166      145 TXOID: .ASCII /Overlay: /<200>
73 000413      104      145      166 TXDEV: .ASCII /Device name: /<200>
74 000431      040      075      040 EQSTR: .ASCII / = /<200>
75 000435      040      040      040 SPSTR: .ASCII / /<200>
76 000443      014      200      FF: .BYTE 14, 200 ;Form feed
77 000445      015      012      200 CRLF: .BYTE CR, LF, 200 ;Carriage return / line feed
78 000450      124      123      130 OVLREG: .ASCIZ /TSX-Plus overlay regions/
79 000501      123      171      163 STKSS: .ASCIZ /System /<200>
80 000512      111      156      164 STKINT: .ASCIZ /Interrupt /<200>
81 000526      105      115      124 STKEMT: .ASCIZ /EMT /<200>
82 000534      123      164      141 STKCNT: .ASCIZ /Stack Contents/
83 000553      040      057      040 STKDLM: .ASCII \ / \<200>
84 000557      111      156      166 INVSTK: .ASCIZ /Invalid stack pointer/
85 000605      111      156      164 INTLEV: .ASCIZ /Interrupt Level/
86 000625      106      157      162 FRKLEV: .ASCIZ /Fork Level/
87 000640      123      171      163 STKOV: .ASCIZ /System stack overflow/
88 000666      112      157      142 CURJOB: .ASCII /Job executing at time of dump/<200>
89      .LIST BEX
90      .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 XON/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 000724   DODUMP:
28
29         ; Disable interrupt and locate the dump device registers.
30
31         ;
32 000724   012737   001140' 000004   ; DISABL          ; Disable all interrupts
33         ; MOV        #DMPTRP,@#4    ; Capture the trap 4 address
34 000732   013705   000000G   ; MOV        #340,@#6    ; Retain disable interrupt status
35 000736   042705   000007    ; MOV        VDMTCR,R5   ; Find the display device CSR
36 000742   005015    ; BIC        #7,R5       ; Adjust for receiver CSR
37 000744   103411    ; CLR        (R5)        ; Check for receiver CSR
38 000746   010537   000050'   ; BCS        1$          ; Br if receiver not available
39 000752   010537   000046'   ; MOV        R5,RCSR     ; Insert receiver address
40 000756   062737   000002   000046' ; MOV        R5,RBUF     ; Insert receiver buffer
41 000764   117700   177056    ; ADD        #2,RBUF     ; offset 2 from CSR
42 000770   062705   000004    ; MOV        @RBUF,R0    ; Throw away any pending input
43 000774   005015    ; ADD        #4,R5       ; Adjust to transmit CSR
44 000776   103457    ; CLR        (R5)        ; Check for transmitter CSR
45 001000   010537   000044'   ; BCS        50$         ; Br if transmitter not available
46 001004   010537   000042'   ; MOV        R5,TCSR     ; Insert transmitter CSR
47 001010   062737   000002   000042' ; MOV        R5,TBUF     ; Insert transmitter buffer address
48         ; ADD        #2,TBUF     ; offset 2 from CSR
49
50         ; Save values on entry (registers, kernel mapping, etc.).
51
52 001016   013737   000000G 000020' ; MOV        @#KPAR6,SVPAR6 ; Save kernel PAR 6
53 001024   013737   000000G 000016' ; MOV        TRPAR5,SVPAR5 ; Save kernel PAR 5
54 001032   012705   000016'   ; MOV        #SVREG,R5   ; Save registers on current stack
55 001036   012700   000006    ; MOV        #6,R0       ; Save registers R0-R6
56 001042   012645    ; MOV        (SP)+,-(R5) ; Save all the register contents
57 001044   077002    ; SOB        R0,10$      ; Continue until all register saved

```

DODUMP -- Dump entry point

```

58      ;
59      ;   Output information concerning the current state.
60      ;
61 001046      PRINT    #FF      ;Output form feed character
62 001056 004737 001150'      CALL    DATTIM      ;Output date and time information
63 001062 004737 001476'      CALL    ERROR      ;Output error status information
64 001066 004737 002414'      CALL    JOB        ;Output current job information
65 001072 004737 001642'      CALL    REG        ;Output the register contents
66 001076 004737 001666'      CALL    MODADR      ;Output TSX-Plus module addresses
67 001102 004737 002012'      CALL    OVRADR      ;Output TSX-Plus overlay regions
68 001106 004737 002314'      CALL    MISC        ;Output miscellaneous information
69 001112 004737 002474'      CALL    LINE        ;Output line table information
70 001116 004737 002506'      CALL    DEVICE      ;Output handler table information
71 001122 004737 002106'      CALL    STACK      ;Output the stack contents
72 001126 004737 002520'      CALL    FORK        ;Output fork block information
73 001132 004737 002556'      CALL    JCBINF      ;Output job context information
74 001136 000000      50$:    HALT      ;Now where to?
75      ;
76      ;   Trap to 4 control.
77      ;
78 001140 052766 000001 000002 DMPTRP: BIS    #1,2(SP)      ;Buffer not found, set c-bit
79 001146 000002      RTI      ;Return from trap

```

DATTIM -- Display date and time information

```

1          .SBTTL  DATTIM -- Display date and time information
2          ;-----
3          ; DATTIM is called to display the current date and time information.
4          ;
5 001150    DATTIM: PRINT    #CRLF          ;Display CR/LF
6 001160    MOV      SYSDAT,R5          ;Get current date
7 001164    BEQ      10$              ;No date set - skip date
8 001166    MOV      R5,R1              ;Copy to working register
9 001170    ASH      #-5,R1             ;Find the day (bits 5-9)
10 001174    BIC      #^C37,R1          ;Mask unwanted bits
11 001200    MOV      #$DEC,R3          ;Force decimal display
12 001204    CALL    OUTVAL             ;Print the day
13 001210    TTYOUT   #'-'             ;Output "-"
14 001220    MOV      R5,R1              ;Copy date to working register
15 001222    ASH      #-9,R1             ;Find the month (bits 10-13)
16 001226    BIC      #^C36,R1          ;Mask unwanted bits (word offset)
17 001232    MOV      MONTH-2(R1),R1    ;Offset into month table
18 001236    MOV      #$RAD50,R3        ;Force rad50 display
19 001242    CALL    OUTRAD             ;Output three char month
20 001246    TTYOUT   #'-'             ;Output "-"
21 001256    MOV      R5,R1              ;Copy date to working register
22 001260    BIC      #^C37,R1          ;Find the year (bits 0-4)
23 001264    ADD      #72,R1            ;Year is relative to 1972
24 001270    MOV      #$DEC,R3          ;Force decimal display
25 001274    CALL    OUTVAL             ;Print the year
26 001300    PRINT    #SPSTR           ;Print blank space
27          ;
28          ; Output the current time.
29          ;
30 001310    10$: MOV      SYTIMH,R0      ;Get the current time (high)
31 001314    MOV      SYTIML,R1          ; clock ticks past midnight (low)
32 001320    MOV      TK1SEC,R3          ;Get number of clock ticks per second
33 001324    CALL    DIV32               ;Divide clock ticks per second
34 001330    MOV      R2,-(SP)           ;Save ticks
35 001332    MOV      #60,R3             ;Get 60. per unit
36 001336    MOV      #2,R5              ;Process seconds, minutes, hours
37 001342    11$: CALL    DIV32           ;Divide by 60. per unit
38 001346    MOV      R2,-(SP)           ;Save remainder
39 001350    SOB      R5,11$             ;Continue for seconds, minutes, hours
40 001352    MOV      R1,-(SP)           ;Save final quotient
41 001354    MOV      #4,R5              ;Loop for minutes, seconds, and ticks
42 001360    BR       21$                ;Output the time past midnight
43 001362    20$: TTYOUT   #'.'           ;Output ":"
44 001372    21$: MOV      (SP)+,R1       ;Get the next value
45 001374    CLR      R0                 ;Clear high order
46 001376    DIV      #10,R0             ;Divide by ten
47 001402    ADD      #'0,R0             ;Convert to ascii number
48 001406    TTYOUT   R0                 ;Display high digit (quotient)
49 001412    ADD      #'0,R1             ;Convert to ascii number
50 001416    TTYOUT   R1                 ;Display low digit (remainder)
51 001424    SOB      R5,20$             ;Continue for hours, minutes, seconds, ticks
52 001426    PRINT    #CRLF             ;Display CR/LF
53 001436    RETURN

```

```

1          .SBTTL  DIV32  -- Divide 16-bit into 32-bit
2          ; -----
3          ; DIV32 divides a 16-bit number in R3 into a 32-bit number
4          ; in R0 (high order) and R1 (low order).  The quotient is
5          ; returned in R0, R1 and the remainder is returned in R2.
6          ; All other registers are preserved.
7          ;
8 001440 010446 DIV32: MOV     R4,-(SP)      ;Save registers
9 001442 005002      CLR     R2             ;Initialize remainder
10 001444 012704 000037 MOV     #31.,R4      ;Initialize shift count
11 001450 006301 1$:  ASL     R1             ;Shift from low order
12 001452 006100      ROL     R0             ; through high order
13 001454 006102      ROL     R2             ; into remainder
14 001456 020203      CMP     R2,R3          ;Check remainder for subtraction
15 001460 103402      BLO     2$             ;Br if unable to subtract
16 001462 160302      SUB     R3,R2          ;Subtract divisor
17 001464 005201      INC     R1             ;Increment quotient
18 001466 005304 2$:  DEC     R4             ;Check remaining bit shift count
19 001470 100367      BPL     1$             ;Continue if more to consider
20 001472 012604      MOV     (SP)+,R4      ;Restore registers
21 001474 000207      RETURN

```

ERROR -- Display error status information

```

1          .SBTTL  ERROR  -- Display error status information
2          ;-----
3          ;  ERROR is called to display the error status information.
4          ;
5 001476    ERROR:  PRINT  #CRLF          ; Display CR/LF
6 001506    PRINT  #DMPTXT        ; Display fatal error message
7 001516    PRINT  #TXARG         ; Display "Arg. value"
8 001526    013701  000000G      MOV  DIEARG,R1      ; Get the argument value
9 001532    005003          CLR  R3              ; Set display characteristics
10 001534    004737  004024'     CALL  OUTVAL        ; Output argument value
11 001540    PRINT  #CRLF          ; Display CR/LF
12 001550    013705  000000G      MOV  DMPOVL,R5     ; Get the rad50 overlay identifier
13 001554    001413          BEQ  1$             ; Br if zero, not in overlay
14 001556    PRINT  #TXOID        ; Display "Overlay: "
15 001566    010501          MOV  R5,R1            ; Copy overlay identifier
16 001570    004737  004220'     CALL  OUTRAD        ; Output rad50 overlay identifier
17 001574    PRINT  #CRLF          ; Display CR/LF
18 001604    013705  000000G      1$: MOV  DMPHND,R5  ; Get the rad50 handler identifier
19 001610    001413          BEQ  2$             ; Br is zero, not in handler
20 001612    PRINT  #TXDEV        ; Display "Device name: "
21 001622    010501          MOV  R5,R1            ; Copy device handler identifier
22 001624    004737  004220'     CALL  OUTRAD        ; Output rad50 handler identifier
23 001630    PRINT  #CRLF          ; Display CR/LF
24 001640    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 001642    REG:    PRINT    #CRLF          ;Display CR/LF
6 001652    CLR     R2          ;Clear offset pointer
7 001654    MOV     #PRTREG,R5      ;Point to register output table
8 001660    CALL    OLIST          ;Output the list of values
9 001664    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 001666    MODADR:
6 001666
7 001676    012737  000000G 000022'    PRINT  #CRLF          ;Display CR/LF
8 001704    012737  000000G 000024'    MOV      #TSGEN,ADGEN      ;Save the TSGEN address
9 001712    012737  000000G 000026'    MOV      #TSTIO,ADTIO      ;Save the TSTIO address
10 001720    012737  000000G 000030'   MOV      #TSEXC,ADEXEC     ;Save the TSEXC address
11 001726    012737  000000G 000032'   MOV      #TSEMT,ADEMT      ;Save the TSEMT address
12 001734    013705  000000G          MOV      #TSINIT,ADINIT     ;Save the TSINIT address
13 001740    072527  000006          MOV      TIOBAS,R5          ;Get PAR of TSTIOX
14 001744    010537  000034'          ASH      #6,R5             ;Convert to address
15 001750    013705  000000G          MOV      R5,ADTIOX          ;Get the TSTIOX address
16 001754    072527  000006          MOV      LOKBAS,R5          ;Save PAR of TSLOCK
17 001760    010537  000036'          ASH      #6,R5             ;Convert to address
18 001764    013705  000000G          MOV      R5,ADLOCK          ;Save the TSLOCK address
19 001770    072527  000006          MOV      CSHBAS,R5          ;Get PAR of TSCASH
20 001774    010537  000040'          ASH      #6,R5             ;Convert to address
21 002000    012705  000100'          MOV      R5,ADCASH          ;Save the TSCASH address
22 002004    004737  002702'          MOV      #ADDR,R5          ;Point to address output list
23 002010    000207          CALL    OLIST          ;Output address contents
          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 --> $OVTAB:
8          ;          .WORD  <IDENTIFIER>, <KPAR5>, <WORD COUNT>
9          ;          DUMMY SUBROUTINES FOR ALL OVERLAY SEGMENTS
10         ;
11 002012 OVRADR:
12 002012      PRINT  #CRLF          ; Display CR/LF
13 002022      PRINT  #OVLREG        ; Display header
14 002032 013705 000000G      MOV    OVRADD, R5          ; Find the overlay table address
15 002036 012501      10$: MOV    (R5)+, R1              ; Get the overlay identifier
16 002040 004737 004220'      CALL   OUTRAD              ; Display rad50 id
17 002044      PRINT  #EQSTR         ; Display " = "
18 002054 012501      MOV    (R5)+, R1              ; Get the KPAR5 value
19 002056 005003      CLR     R3              ; Default octal display value
20 002060 004737 004024'      CALL   OUTVAL             ; Display KPAR5 value
21 002064 005725      TST    (R5)+          ;
22 002066      PRINT  #CRLF          ; Display CR/LF
23 002076 021527 004537      CMP    (R5), #4537        ; Compare with a <JSR R5, $OVRH> instruction
24 002102 001355      BNE    10$              ; Br if not end of the table
25 002104 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 002106   STACK:  PRINT  #CRLF          ;Display CR/LF
15 002116   MOV     DIESP,R5          ;Get the stack pointer
16 002122   MOV     R5,R4            ;Copy the base of the stack
17 002124   5$:    CMP     R5,#SS      ;Below the top of the system stack?
18 002130   BHI     10$              ;No, SP not in system stack
19         ;
20         ; Stack pointer is in the system stack area.
21         ;
22 002132   SUB     #SS,R4            ;Calculate number of bytes to dump
23 002136   BEQ     50$              ;No stack space used
24 002140   NEG     R4                ;Convert to positive byte count
25 002142   PRINT  #STKSS            ;Display "System Stack Contents"
26 002152   CALL   OSTACK            ;Output the stack contents
27 002156   RETURN
28 002160   10$:   CMP     R5,INTSTK   ;Below the top of the interrupt stack?
29 002164   BHI     20$              ;No, SP not in interrupt stack
30         ;
31         ; Stack pointer is in the interrupt stack area.
32         ;
33 002166   SUB     INTSTK,R4         ;Calculate number of bytes to dump
34 002172   BEQ     50$              ;No stack space used
35 002174   NEG     R4                ;Convert to positive byte count
36 002176   PRINT  #STKINT           ;Display "Interrupt Stack Contents"
37 002206   CALL   OSTACK            ;Output the stack contents
38 002212   RETURN
39 002214   20$:   CMP     R5,#JSTKND  ;Above the end of the context stack?
40 002220   BLO     30$              ;Invalid stack pointer
41 002222   CMP     R5,#JSTK         ;Below the top of the context stack?
42 002226   BHI     30$              ;No, SP not in job context stack
43         ;
44         ; Stack pointer is in the job's context stack.
45         ;
46 002230   21$:   SUB     #JSTK,R4   ;Calculate number of bytes to dump
47 002234   BEQ     50$              ;No stack space used
48 002236   NEG     R4                ;Convert to positive byte count
49 002240   PRINT  #STKEMT           ;Display "EMT Stack Contents"
50 002250   CALL   OSTACK            ;Output the stack contents
51 002254   RETURN
52         ;
53         ; Stack pointer is invalid.  Output 40 words where SP is pointing.
54         ;
55 002256   30$:   PRINT  #INVSTK     ;Display "Invalid stack"
56 002266   CMP     R5,#VPAR5        ;Check for EMT stack overflow
57 002272   BLO     40$              ;Br if below 120000, unknown stack

```

STACK -- Display contents of stack

58	002274	012705	000000G		MOV	#VPAR6,R5	; Adjust stack pointer to EMT stack
59	002300	000753			BR	21\$	; Dump job context stack
60	002302	012704	000310	40\$:	MOV	#100.*2,R4	; Dump 100. words from stack pointer
61	002306	004737	003036'		CALL	OSTACK	; Output the stack contents
62	002312	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 002314    MISC:
6 002314
7 002324    105737  00000006      PRINT  #CRLF          ;Display CR/LF
8 002330    002404          TSTB   INTLVL          ;Check interrupt level
9 002332          BLT      10$          ;Br if not executing interrupt
10 002342    105737  00000006      PRINT  #INTLEV         ;Display "Interrupt level"
11 002346    001404          TSTB   FRKPRI          ;Check fork priority
12 002350          BEQ      20$          ;Br if not executing fork
13 002360    023727  00000006 123456 20$:  PRINT  #FRKLEV         ;Display "Fork level"
14 002366    001404          CMP     SEND,#123456      ;Check end of system stack
15 002370          BEQ      30$          ;Br if stack end is valid
16 002400    005002          PRINT  #STKOVN          ;Display "System stack overflow"
17 002402    012705  000164'      CLR     R2          ;Clear offset pointer
18 002406    004737  002702'      MOV     #VMISC,R5      ;Point to miscellaneous output list
19 002412    000207          CALL    OLIST          ;Output the list of values
          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 002414    JOB:  PRINT  #CRLF          ;Display CR/LF
6 002424          PRINT  #CURJOB        ;Disp "Job executing at time of dump"
7 002434          PRINT  #EQSTR         ;Display " = "
8 002444    113701 000000G          MOV  CORUSR,R1      ;Get the current operating job index
9 002450    006201          ASR  R1        ;Divide by two for the job number
10 002452    012703 000002          MOV  #$DEC,R3       ;Force decimal display
11 002456    004737 004024'        CALL  OUTVAL        ;Output the job number
12 002462          PRINT  #CRLF         ;Display CR/LF
13 002472    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 002474  012705  000622'  LINE:  MOV      #JLIN,R5      ;Point to job line table
6 002500  004737  003112'      CALL    OTABLE        ;Output line definition values
7 002504  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 002506  012705  001410'  DEVICE: MOV      #DHAN,R5      ;Point to device table
6 002512  004737  003112'      CALL      OTABLE      ;Output device table values
7 002516  000207
8          RETURN

```

FORK -- Display fork block information

```

1          .SBTTL FORK          -- Display fork block information
2          ;-----
3          ; FORK is called to display information on the FORK list.
4
5 002520 013702 0000000 FORK:  MOV    FRKQGE,R2          ;Get pointer to pending FORK requests
6 002524 001413          BEQ     2$              ;Br if no FORK pending
7 002526          1$:  PRINT   #CRLF            ;Display CR/LF
8 002536 012705 001546'  MOV     #FRKREQ,R5        ;Point to fork display list
9 002542 004737 002702'  CALL    OLIST           ;Output the list of values
10 002546 016202 0000000 MOV     FQ$LNK(R2),R2      ;Get next FORK element in link list
11 002552 001365          BNE     1$              ;Br if more to display
12 002554 000207          2$:  RETURN

```

```

1          . SBTTL JCBINF -- Display job context information
2          ; -----
3          ; JCBINF is called to display job context information.
4          ;
5 002556 113700 000000G JCBINF: MOVB CORUSR,R0 ;Get the current executing user
6 002562 001446 BEQ 10$ ;Br if no user is active
7 002564 005760 000000G TST LBASE(R0) ;Test base page no. of job context mapping
8 002570 001443 BEQ 10$ ;Br if job context not mapped
9 002572 016001 000000G MOV LCXPAR(R0),R1 ;Get PAR relocation of job context
10 002576 001440 BEQ 10$ ;Br if job context not mapped
11 002600 032760 000000G 000000G BIT #$INCOR,LSW(R0) ;Is user job in memory?
12 002606 001434 BEQ 10$ ;Br if not in memory
13 002610 013737 000000G 000054' MOV @KPAR6,SAVKP6 ;Save current par6 mapping
14 002616 010137 000000G MOV R1,@KPAR6 ;Map to context region
15          ;
16          ; Convert translated function code back to recognizable form.
17          ;
18 002622 113700 000001G MOVB EMTBLK+1,R0 ;Get EMT function code
19 002626 042700 177400 BIC #C377,R0 ;Clear sign extend bits
20 002632 120027 000000G CMPB R0,#RT11EX ;Is this RT-11 or TSX function
21 002636 103402 BLO 5$ ;Br if Rt-11 function
22 002640 062700 000000C ADD #C100-RT11EX,R0 ;Add table bias
23 002644 110037 000056' 5$: MOVB R0,EMTFUN ;Store in local cell
24          ;
25          ; Display information from job context region.
26          ;
27 002650 PRINT #CRLF ;Display carriage return/line feed
28 002660 012705 001646' MOV #VJCB,R5 ;Point to job context output list
29 002664 005002 CLR R2 ;Clear the offset pointer
30 002666 004737 002702' CALL OLIST ;Output job context information
31 002672 013737 000054' 000000G MOV SAVKP6,@KPAR6 ;Restore kernel par 6 mapping
32 002700 000207 10$: RETURN

```

OLIST -- Output a list of information

```

1          .SBTTL  OLIST  -- Output a list of information
2          ;-----
3          ; Output a list of information.
4          ;
5          ; Inputs:
6          ;   R5 = points to the head of the list of output information
7          ;   R2 = pointer to the offset (index) to the data item
8          ;
9 002702 010546 OLIST:  MOV    R5,-(SP)      ;Save registers
10 002704 010446      MOV    R4,-(SP)
11 002706 010346      MOV    R3,-(SP)
12 002710 010246      MOV    R2,-(SP)
13 002712 010146      MOV    R1,-(SP)
14          ;
15          ; Display the list header
16          ;
17 002714 012501      MOV    (R5)+,R1      ;Point to list header
18 002716 001437      BEQ     100$          ;Br if not header
19 002720      PRINT    ;Display the list header
20 002724 000434      BR      100$          ;Output entire display list
21          ;
22          ; Display identifier
23          ;
24 002726 10$: PRINT    (R5)                ;Print identifier.
25 002734 30$: PRINT    #EQSTR              ;Point to string " = "
26 002744 016503 000002  MOV    2(R5),R3    ;Get field attributes
27 002750 016504 000004  MOV    4(R5),R4    ;Get the field address
28 002754 060204      ADD     R2,R4         ;Add field bias
29 002756 032703 000001  BIT     ##BYTE,R3  ;Word or byte storage element?
30 002762 001404      BEQ     31$           ;Br if word element
31 002764 111401      MOVB    @R4,R1        ;Get numeric value
32 002766 042701 177400  BIC     #^C377,R1  ;Clear sign extend
33 002772 000401      BR      32$           ;Continue
34 002774 011401 31$: MOV    @R4,R1        ;Get the numeric value
35 002776 004737 004024' 32$: CALL  OUTVAL   ;Output the numeric value
36 003002      PRINT    #CRLF              ;Point to carriage return / line feed
37 003012 062705 000006 50$: ADD     #6,R5   ;Increment to next list entry
38          ;
39          ; Continue output of list elements.
40          ;
41 003016 005715 100$: TST     (R5)          ;Point to the next list entry
42 003020 001342      BNE     10$           ;Br if list continues
43          ;
44          ; Finished
45          ;
46 003022 012601      MOV     (SP)+,R1      ;Restore registers
47 003024 012602      MOV     (SP)+,R2
48 003026 012603      MOV     (SP)+,R3
49 003030 012604      MOV     (SP)+,R4
50 003032 012605      MOV     (SP)+,R5
51 003034 000207      RETURN                ;Finished

```

OSTACK -- Display stack contents

```

1          .SBTTL  OSTACK  -- Display stack contents
2          ; -----
3          ;   Output the stack contents.
4          ;
5          OSTACK:
6          ASR      R4              ; Convert byte count to word count
7          PRINT    #STKCNT         ; Display header
8          CLR      R3              ; Set for octal display
9          MOV      R5,R1           ; Copy the stack address
10         CALL     OUTVAL           ; Display stack address
11         PRINT    #STKDLN         ; Display delimiter
12         MOV      (R5)+,R1         ; Get the stack contents
13         CALL     OUTVAL           ; Display stack contents
14         PRINT    #CRLF            ; Display CR/LF
15         SOB      R4,10$           ; Continue until done
16         RETURN

```

OTABLE -- Output tables of information

```

1          .SBTTL  OTABLE  -- Output tables of information
2          ;-----
3          ;  Output tables of information.
4          ;
5 003112    OTABLE:
6 003112    004737    003150'    CALL    TINIT          ; Initialize for table output
7 003116    004737    003204'    1$:    CALL    TCLR          ; Clear output tables
8 003122    004737    003354'    CALL    TACCR          ; Accrue table information
9 003126    004737    003630'    CALL    TOUT          ; Display table values
10 003132    005715          TST    (R5)          ; Check for more table information
11 003134    001370          BNE    1$          ; Br if more to display
12 003136          PRINT  #CRLF          ; Display CR/LF
13 003146    000207          RETURN

```

OTABLE -- Output tables of information

```

1      ;
2      ; Initialize for display start or continuation of table format.
3      ;
4 003150 TINIT: PRINT #CRLF ;Display CR/LF
5 003160 PRINT (R5)+ ;Display the table title
6 003166 005037 000052' CLR COL ;Clear column counter
7 003172 PRINT #CRLF ;Display CR/LF
8 003202 000207 RETURN
9      ;
10     ; Clear table entries for continuation of table format.
11     ;
12 003204 012702 000014 TCLR: MOV #NCOL,R2 ;Initilize table storage information
13 003210 006302 ASL R2 ;Convert to word index
14 003212 005062 000060' 10$: CLR TSCOL(R2) ;Clear starting column number
15 003216 005062 000112' CLR TECOL(R2) ;Clear ending column number
16 003222 005062 000144' CLR TSINX(R2) ;Clear starting index number
17 003226 005062 000176' CLR TEINX(R2) ;Clear ending index number
18 003232 005062 000230' CLR TATT(R2) ;Clear attribute flags
19 003236 005062 000262' CLR TADR(R2) ;Clear address location
20 003242 162702 000002 SUB #2,R2 ;Subtract from table index
21 003246 003361 BGT 10$ ;Continue for all entries
22 003250 002427 BLT 20$ ;All entries are clear
23 003252 005737 000052' TST COL ;Check column counter
24 003256 001755 BEQ 10$ ;Br if first table initialization call
25     ;
26     ; Table continuation display.
27     ;
28 003260 PRINT #CRLF ;Print CR/LF
29 003270 PRINT #CRLF ;Print CR/LF
30 003300 PRINT TADR(R2) ;Print columnar heading
31 003310 016201 000112' MOV TECOL(R2),R1 ;Find ending column information
32 003314 004737 004342' CALL BLKPAD ;Blank fill
33 003320 PRINT #SPSTR ;Print blanks for spacing
34 003330 000207 20$: RETURN

```

OTABLE -- Output tables of information

```

1
2      ; Accrue information concerning table display while printing column headings.
3
4 003332      ; NXTCOL:
5 003332 016562 000004 000262'      MOV 4(R5),TADR(R2) ;Store print string address
6 003340 062705 000012      ADD #10.,R5 ;Offset to the next column entry
7 003344      PRINT #SPSTR ;Print blanks for spacing
8 003354 062702 000002      TACCR: ADD #2,R2 ;Increment the field index
9 003360 020227 000014      CMP R2,#NCOL ;End of columns allowed?
10 003364 002120      BGE 50$ ;Br is no more column storage
11 003366 005715      TST (R5) ;See if a column entry exists
12 003370 001516      BEQ 50$ ;End of table found
13 003372 013762 000052' 000060'      MOV COL,TSCOL(R2) ;Point to starting field position
14 003400      PRINT (R5) ;Display the column heading
15 003406 013762 000052' 000112'      MOV COL,TECOL(R2) ;Set ending column position
16 003414 016562 000002 000230'      MOV 2(R5),TATT(R2) ;Store field attributes
17 003422 016500 000006      MOV 6(R5),R0 ;Get the starting index number
18 003426 001002      BNE 1$ ;Br if start is specified
19 003430 013700 000144'      MOV TSINX,R0 ;Set as starting table number
20 003434 010062 000144'      1$: MOV R0,TSINX(R2) ;Store starting index number
21 003440 016500 000010      MOV 10(R5),R0 ;Get the ending index number
22 003444 001002      BNE 2$ ;Br if end is specified
23 003446 013700 000176'      MOV TEINX,R0 ;Set as ending table number
24 003452 010062 000176'      2$: MOV R0,TEINX(R2) ;Store ending index number
25 003456 012700 000006      MOV #6,R0 ;Assume 6 digit (word element)
26 003462 032762 000001 000230'      BIT ##BYTE,TATT(R2) ;Check field attributes
27 003470 001402      BEQ 10$ ;Br if word element
28 003472 012700 000003      MOV #3,R0 ;Assume 3 digit (byte element)
29 003476 066200 000060'      10$: ADD TSCOL(R2),R0 ;Add starting position
30 003502 026200 000112'      CMP TECOL(R2),R0 ;Determine if ending position
31 003506 001427      BEQ 30$ ;Field is exactly enough
32 003510 101013      BHI 20$ ;Br if field too long
33 003512 010001      MOV R0,R1 ;Copy ending column address
34 003514 166201 000112'      SUB TECOL(R2),R1 ;Find difference
35 003520 010062 000112'      MOV R0,TECOL(R2) ;Force ending column longer
36 003524 112700 000040      MOVB #' ,R0 ;Store blank
37 003530      11$: TTYOUT ;Pad output with blanks
38 003534 077103      SOB R1,11$ ;Continue until padded
39 003536 000413      BR 30$ ;Go on to the next entry
40 003540 166200 000112'      20$: SUB TECOL(R2),R0 ;Subtract the ending column
41 003544 032762 040000 000230'      BIT ##RJUST,TATT(R2) ;Check for right justify
42 003552 001403      BEQ 21$ ;Br if left justify
43 003554 160062 000060'      SUB R0,TSCOL(R2) ;Adjust starting column position
44 003560 000402      BR 30$
45 003562 060062 000112'      21$: ADD R0,TECOL(R2) ;Adjust ending column position
46 003566 026227 000112' 000120 30$: CMP TECOL(R2),#80. ;Check for line overflow
47 003574 103656      BLO NXTCOL ;Continue collecting columns
48 003576 005062 000060'      CLR TSCOL(R2) ;Clear partial entry
49 003602 005062 000112'      CLR TECOL(R2)
50 003606 005062 000144'      CLR TSINX(R2)
51 003612 005062 000176'      CLR TEINX(R2)
52 003616 005062 000230'      CLR TATT(R2)
53 003622 005062 000262'      CLR TADR(R2)
54 003626 000207      50$: RETURN

```

OTABLE -- Output tables of information

```

1      ;
2      ;   Display index value and data.
3      ;
4
5 003630 010446      TOUT:  MOV     R4,-(SP)      ; Save some registers
6 003632 010546      MOV     R5,-(SP)
7
8      ;   Display index offset into table.
9      ;
10     MOV     TSINX,R5      ; Find the starting index
11 003640      1$:  PRINT    #CRLF      ; Display CR/LF
12 003650      MOV     R5,R1      ; Copy index number
13 003652 010501      CLR     R2      ; Clear column index
14 003654 016203 000230'  MOV     TATT(R2),R3      ; Numeric attribute
15 003660 004737 004024'  CALL    OUTVAL      ; Display index value
16 003664 000430      BR      40$      ; Enter column output stream
17
18      ;   Determine if the value should be displayed.
19      ;
20 003666 020562 000144'  10$:  CMP     R5,TSINX(R2)      ; Is the current index < starting
21 003672 002425      BLT     40$      ; Yes, skip entry
22 003674 020562 000176'  CMP     R5,TEINX(R2)      ; Is the current index > ending
23 003700 003022      BGT     40$      ; Yes, skip entry
24
25      ;   Output rows of information.
26      ;
27 003702 004737 004342'      CALL    BLKPAD      ; Pad with spaces
28 003706 016203 000230'      MOV     TATT(R2),R3      ; Get the attribute
29 003712 016201 000262'      MOV     TADR(R2),R1      ; Check for existing entry
30 003716 060501      ADD     R5,R1      ; Add the index offset (byte)
31 003720 032703 000001      BIT     #$BYTE,R3      ; Check for byte storage
32 003724 001404      BEQ     20$      ; Br if word storage
33 003726 111101      MOVB    (R1),R1      ; Get the stored byte value
34 003730 042701 177400      BIC     #^C377,R1      ; Kill sign extend bits
35 003734 000402      BR      30$      ; Continue
36 003736 060501      20$:  ADD     R5,R1      ; Add the index offset (word)
37 003740 011101      MOV     (R1),R1      ; Get the stored word value
38 003742 004737 004024'  30$:  CALL    OUTVAL      ; Output the value
39 003746 062702 000002      40$:  ADD     #2,R2      ; Increment the index
40 003752 016201 000060'  MOV     TSCOL(R2),R1      ; Get the starting column
41 003756 001343      BNE     10$      ; End of table entries
42 003760 005205      INC     R5      ; Increment the index offset
43 003762 020537 000176'  CMP     R5,TEINX      ; Check for ending index
44 003766 101724      BLOS    1$      ; Br if more tables to output
45 003770 012605      MOV     (SP)+,R5      ; Restore registers
46 003772 012604      MOV     (SP)+,R4
47 003774 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 003776   OUTSTR:
10 003776 112100 1$:      MOVB      (R1)+,R0          ;Get next output byte
11 004000 001404      BEQ        10$              ;Zero, output CR / LF
12 004002 100407      BMI        20$              ;Exit on end of output string
13 004004      TTYOUT          ;Output the character
14 004010 000772      BR         1$              ;Contine until end of string
15 004012      10$:      PRINT    #CRLF          ;Display carriage return / line feed
16 004022 000207      20$:      RETURN

```

```

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 004024 010546      OUTVAL: MOV      R5,-(SP)      ; Save registers
10 004026 010446      MOV      R4,-(SP)
11 004030 010346      MOV      R3,-(SP)
12 004032 010246      MOV      R2,-(SP)
13 004034 032703 000004      BIT      $$RAD50,R3      ; Check for RAD50 value
14 004040 001403      BEQ      1$      ; Br if not RAD50
15 004042 004737 004220'      CALL     OUTRAD      ; Output RAD50
16 004046 000420      BR      30$      ; Continue
17 004050 012704 000010      1$:  MOV      #10,R4      ; Assume octal base
18 004054 032703 000002      BIT      $$DEC,R3      ; Check for decimal base
19 004060 001402      BEQ      10$      ; Br if numeric display in octal
20 004062 012704 000012      MOV      #10,R4      ; Set for decimal base
21 004066 012702 000006      10$: MOV      #6,R2      ; Assume word value
22 004072 032703 000001      BIT      $$BYTE,R3      ; Check for byte storage
23 004076 001402      BEQ      20$      ; Br if storage is word value
24 004100 012702 000003      MOV      #3,R2      ; Otherwise byte value
25 004104 004737 004122'      20$: CALL     CONVAL      ; Convert numeric value
26 004110 012602      30$:  MOV      (SP)+,R2      ; Restore registers
27 004112 012603      MOV      (SP)+,R3
28 004114 012604      MOV      (SP)+,R4
29 004116 012605      MOV      (SP)+,R5
30 004120 000207      RETURN
31
32 004122      ;
33 004122 005302      CONVAL: DEC      R2      ; Say one more digit converted
34 004124 002411      BLT      2$      ; Br if all display digits converted
35 004126 005000      CLR      R0      ; Clear high order result
36 004130 071004      DIV      R4,R0      ; Divide number by numeric base
37 004132 062701 000060      ADD      #60,R1      ; Convert remainder to ascii digit
38 004136 010146      MOV      R1,-(SP)      ; Save ascii digit on stack
39 004140 010001      MOV      R0,R1      ; Set result register
40 004142 001406      BEQ      3$      ; Br if no more digits to convert
41 004144 004737 004122'      CALL     CONVAL      ; Call conversion for next digit
42 004150 112600      2$:  MOVB     (SP)+,R0      ; Get the next digit to display
43 004152      TTYOUT      ; Output the ascii character
44 004156 000207      RETURN
45 004160 005702      3$:  TST      R2      ; Check field output size
46 004162 001772      BEQ      2$      ; No pad necessary
47 004164 032703 040000      BIT      $$RJUS,R3      ; Check for right justify
48 004170 001767      BEQ      2$      ; Br if not right justify
49 004172 112700 000040      MOVB     #' ,R0      ; Default pad to blank
50 004176 032703 020000      BIT      $$ZRPAD,R3      ; Check for zero pad
51 004202 001402      BEQ      4$      ; Br if blank pad chosen
52 004204 112700 000060      MOVB     #'0,R0      ; Change to zero pad
53 004210      4$:  TTYOUT      ; Output the character
54 004214 077203      SOB      R2,4$      ; Continue until all output
55 004216 000754      BR      2$      ; Display remaining characters

```

OUTRAD -- Convert and print a RAD50 value

```

1          .SBTTL  OUTRAD  -- Convert and print a RAD50 value
2          ;-----
3          ;  OUTRAD - Convert and output rad50 data.
4          ;      R1      = Numeric rad50 value
5          ;
6 004220 010146      OUTRAD: MOV      R1,-(SP)
7 004222 005000      CLR      R0          ;Clear high order
8 004224 071027 003100      DIV      #50*50,R0      ;Divide for 1st byte
9 004230          TTYOUT    R50CHR(R0)      ;Display rad50 character
10 004240 005000      CLR      R0          ;Clear high order
11 004242 071027 000050      DIV      #50,R0      ;Divide for 2nd byte
12 004246          TTYOUT    R50CHR(R0)      ;Display rad50 character
13 004256          TTYOUT    R50CHR(R1)      ;Display rad50 character
14 004266 012601      MOV      (SP)+,R1
15 004270 000207      RETURN
16          ;
17 004272          040      101      102      R50CHR: .ASCII / ABCDEFGHIJKLMNOPQRSTUVWXYZ$. 0123456789/
18          004275          103      104      105
19          004300          106      107      110
20          004303          111      112      113
21          004306          114      115      116
22          004311          117      120      121
23          004314          122      123      124
24          004317          125      126      127
25          004322          130      131      132
26          004325          044      056      040
27          004330          060      061      062
28          004333          063      064      065
29          004336          066      067      065
30          004341          071

```

. EVEN

```

18          .SBTTL  BLKPAD  -- Blank pad output stream
19          ;-----
20          ;  BLKPAD - Blank pad ouput stream.
21          ;      R1      = ending column number
22          ;
23          BLKPAD: MOVB     #' ,R0          ;Initialize ouput to space
24          SUB      COL,R1          ;Figure number of blanks
25          BLE      20$          ;Br if already past column
26          10$:      TTYOUT    ;Display space
27          SOB      R1,10$        ;Continue
28          20$:      RETURN
29
30

```

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 004364 005737 000050'  OUTCHR: TST      RCSR      ; Check for receiver buffers
7 004370 001415          BEQ      OPRINT      ; No receiver CSR, transmit character
8 004372          OTERM:
9 004372 105777 173452 10$: TSTB      @RCSR      ; Check input port
10 004376 100006          BPL      20$         ; Br if no input pending
11 004400 117737 173442 000344'  MOVB      @RBUF,XFLG  ; Input the character
12 004406 142737 177600 000344'  BICB      #^C177,XFLG  ; Strip eighth bit
13 004414 123727 000344' 000021 20$: CMPB      XFLG,#CTRLQ  ; Check for XON
14 004422 001363          BNE      10$         ; Br if not XON state active
15 004424          OPRINT:
16 004424 105777 173414 30$: TSTB      @TCSR      ; Check transmit done flag
17 004430 100375          BPL      30$         ; Wait for transmit done
18 004432 110077 173404          MOVB      RO,@TBUF  ; Output to transmitter
19 004436 120027 000015          CMPB      RO,#CR      ; Is character a carriage return?
20 004442 001003          BNE      40$         ; Br if not carriage return
21 004444 005037 000052'  CLR      COL      ; Clear column counter
22 004450 000405          BR      50$         ; Finished
23 004452 120027 000040 40$: CMPB      RO,#40      ; Is character printable?
24 004456 103402          BLO      50$         ; Br if not printable
25 004460 005237 000052'  INC      COL      ; Increment column position
26 004464 000207 50$: RETURN          ; Finished
27
28          000001          .END

```

Errors detected: 0

*** Assembler statistics

```

Work file reads: 0
Work file writes: 0
Size of work file: 9950 Words ( 39 Pages)
Size of core pool: 18176 Words ( 71 Pages)
Operating system: RT-11

```

```

Elapsed time: 00:01:18.01
,LP:TSDUMP=DK:TSDUMP/C/N:SYM

```

\$BYTE	5-22#	10-10	10-11	10-12	10-13	10-17	10-18	10-19	10-24	10-25	10-27	10-30
	10-31	10-32	10-33	10-34	10-35	10-37	10-38	10-42	10-43	10-50	10-52	11-5
	12-5	13-14	14-9	14-10	30-29	34-26	35-31	37-22				
\$DEC	5-23#	11-5	12-5	17-11	17-24	25-10	37-18					
\$INCOR	2-20	29-11										
\$RAD50	5-24#	11-25	11-26	12-6	17-18	37-13						
\$RJUST	5-19#	11-6	11-7	11-8	11-9	11-10	11-11	11-12	11-13	11-14	11-15	11-16
	11-17	11-18	11-19	11-20	11-21	11-22	11-23	11-24	11-27	11-28	11-29	11-30
	11-31	11-32	11-33	11-34	11-35	11-36	11-37	11-38	11-39	11-40	11-41	12-7
	12-8	12-9	12-10	12-11	12-12	12-13	34-41	37-47				
\$ZRPAD	5-20#	11-6	11-7	11-8	11-9	11-10	11-11	11-12	11-13	11-14	11-15	11-16
	11-17	11-18	11-19	11-20	11-21	11-22	11-23	11-24	11-27	11-28	11-29	11-30
	11-31	11-32	11-33	11-34	11-35	11-36	11-37	11-38	11-39	11-40	11-41	12-11
	12-12	12-13	37-50									
ADCASH	9-13	9-13	15-36#	21-20*								
ADDR	9-5#	21-21										
ADEMT	9-9	9-9	15-32#	21-10*								
ADEXEC	9-8	9-8	15-31#	21-9*								
ADGEN	9-6	9-6	15-29#	21-7*								
ADINIT	9-10	9-10	15-33#	21-11*								
ADLOCK	9-12	9-12	15-35#	21-17*								
ADTIO	9-7	9-7	15-30#	21-8*								
ADTIOX	9-11	9-11	15-34#	21-14*								
ARG	16-61	16-61#	17-5	17-5#	17-13	17-13#	17-20	17-20#	17-26	17-26#	17-43	17-43#
	17-48	17-48#	17-50	17-50#	17-52	17-52#	19-5	19-5#	19-6	19-6#	19-7	19-7#
	19-11	19-11#	19-14	19-14#	19-17	19-17#	19-20	19-20#	19-23	19-23#	20-5	20-5#
	21-6	21-6#	22-12	22-12#	22-13	22-13#	22-17	22-17#	22-22	22-22#	23-14	23-14#
	23-25	23-25#	23-36	23-36#	23-49	23-49#	23-55	23-55#	24-6	24-6#	24-9	24-9#
	24-12	24-12#	24-15	24-15#	25-5	25-5#	25-6	25-6#	25-7	25-7#	25-12	25-12#
	28-7	28-7#	29-27	29-27#	30-19	30-19#	30-24	30-24#	30-25	30-25#	30-36	30-36#
	31-7	31-7#	31-11	31-11#	31-14	31-14#	32-12	32-12#	33-4	33-4#	33-5	33-5#
	33-7	33-7#	33-28	33-28#	33-29	33-29#	33-30	33-30#	33-33	33-33#	34-7	34-7#
	34-14	34-14#	34-37	34-37#	35-11	35-11#	36-13	36-13#	36-15	36-15#	37-43	37-43#
	37-53	37-53#	38-9	38-9#	38-12	38-12#	38-13	38-13#	38-28	38-28#		
ARGS	8-5	8-5#	9-5	9-5#	10-5	10-5#	11-5	11-5#	12-5	12-5#	13-4	13-4#
	14-6	14-6#										
BLKPAD	33-32	35-27	38-25#									
CHNADR	14-18	14-18										
CLKPC	10-6	10-6										
CLKPS	10-7	10-7										
COL	15-46#	33-6*	33-23	34-13	34-15	38-26	39-21*	39-25*				
CONVAL	37-25	37-32#	37-41									
CORUSR	2-14	25-8	29-5									
CR	15-7#	15-77	39-19									
CRLF	15-77#	17-5	17-52	19-5	19-11	19-17	19-23	20-5	21-6	22-12	22-22	23-14
	24-6	25-5	25-12	28-7	29-27	30-36	31-14	32-12	33-4	33-7	33-28	33-29
	35-11	36-15										
CSHBAS	10-8	10-8	21-18									
CTRLQ	15-9#	15-65	39-13									
CTRLS	15-10#											
CUREMT	14-8	14-8										
CURFRK	10-9	10-9										
CURJOB	15-88#	25-6										
CXBJOB	10-10	10-10										
CXBOWN	10-11	10-11										
DASH	15-11#											

JSTKND	2-9	23-39							
KPAR6	2-13	16-51	29-13	29-14*	29-31*				
LACTIV	11-24	11-24							
LAFSIZ	11-34	11-34							
LBASE	11-18	11-18	29-7						
LBSPRI	11-30	11-30							
LCDTYP	11-27	11-27							
LCLUNT	11-31	11-31							
LCMPL	11-36	11-36							
LCXPAR	2-22	29-9							
LEMTPC	11-39	11-39							
LF	15-8#	15-77							
LFWLIM	11-35	11-35							
LINCNT	11-23	11-23							
LINE	16-69	26-5#							
LIOCNT	11-22	11-22							
LIOHLD	11-33	11-33							
LJSW	11-38	11-38							
LMEMIN	11-32	11-32							
LNBLKS	11-20	11-20							
LNPRIM	11-28	11-28							
LNSBLK	11-21	11-21							
LOKBAS	10-22	10-22	21-15						
LOKSWP	10-23	10-23							
LPARBS	11-19	11-19							
LPARNT	11-29	11-29							
LPRG1	11-25	11-25							
LPRG2	11-26	11-26							
LSPND	11-40	11-40							
LSTATE	11-17	11-17							
LSW	11-6	11-6	29-11						
LSW10	11-15	11-15							
LSW11	11-16	11-16							
LSW2	11-7	11-7							
LSW3	11-8	11-8							
LSW4	11-9	11-9							
LSW5	11-10	11-10							
LSW6	11-11	11-11							
LSW7	11-12	11-12							
LSW8	11-13	11-13							
LSW9	11-14	11-14							
LSWPBK	11-37	11-37							
LXCL	11-41	11-41							
MAPUSR	10-24	10-24							
MAXDEV	2-14	12-5							
MEM256	10-25	10-25							
MEMSWP	10-26	10-26							
MIOFLG	10-27	10-27							
MISC	16-68	24-5#							
MODADR	16-66	21-5#							
MONTH	15-61#	17-17							
MSGBAS	10-28	10-28							
NCOL	15-12#	15-52	15-53	15-54	15-55	15-56	15-57	33-12	34-9
NDL	2-15	11-28	11-31						
NIOL	2-15	11-31							
NLINES	2-15	11-5							

NMFREQ	10-29	10-29										
NPL	2-15	11-28	11-31	11-41								
NSL	2-15	11-31										
NUMDEV	2-14											
NUMON	10-30	10-30										
NXTCOL	34-4#	34-47										
OLIST	20-8	21-22	24-18	28-9	29-30	30-9#						
OPRINT	39-7	39-15#										
OSTACK	23-26	23-37	23-50	23-61	31-5#							
OTABLE	26-6	27-6	32-5#									
OTERM	39-8#											
OUTBSY	10-31	10-31										
OUTCHR	17-13	17-20	17-43	17-48	17-50	34-37	36-13	37-43	37-53	38-9	38-12	38-13
	38-28	39-6#										
OUTRAD	17-19	19-16	19-22	22-16	37-15	38-6#						
OUTSTR	16-61	17-5	17-26	17-52	19-5	19-6	19-7	19-11	19-14	19-17	19-20	19-23
	20-5	21-6	22-12	22-13	22-17	22-22	23-14	23-25	23-36	23-49	23-55	24-6
	24-9	24-12	24-15	25-5	25-6	25-7	25-12	28-7	29-27	30-19	30-24	30-25
	30-36	31-7	31-11	31-14	32-12	33-4	33-5	33-7	33-28	33-29	33-30	33-33
	34-7	34-14	35-11	36-9#	36-15							
OUTVAL	17-12	17-25	19-10	22-20	25-11	30-35	31-10	31-13	35-15	35-38	37-9#	
OVLREG	15-78#	22-13										
OVRADD	2-19	22-14										
OVRADR	16-67	22-11#										
PNAME	12-6	12-6										
PRTREG	8-5#	20-7										
PSW	2-10											
PVON	10-32	10-32										
R5OCHR	38-9	38-12	38-13	38-17#								
RBUF	15-44#	16-39#	16-40*	16-41	39-11							
RCSR	15-45#	16-38*	39-6	39-9								
REG	16-65	20-5#										
RT11EX	2-23	29-20	29-22									
SAVKP6	15-47#	29-13*	29-31									
SPDJOB	10-33	10-33										
SPSTR	15-75#	17-26	33-33	34-7								
SR3FLG	10-34	10-34										
SS	2-9	23-17	23-22									
SSEND	2-9	24-13										
STACK	16-71	23-14#										
STKCNT	15-82#	31-7										
STKDLM	15-83#	31-11										
STKEMT	15-81#	23-49										
STKINT	15-80#	23-36										
STKLVL	10-35	10-35										
STKQVR	15-87#	24-15										
STKSS	15-79#	23-25										
STRADD	8-5	8-5#	8-6	8-6#	8-7	8-7#	8-8	8-8#	8-9	8-9#	8-10	8-10#
	8-11	8-11#	8-12	8-12#	8-13	8-13#	8-14	8-14#	8-15	8-15#	9-5	9-5#
	9-6	9-6#	9-7	9-7#	9-8	9-8#	9-9	9-9#	9-10	9-10#	9-11	9-11#
	9-12	9-12#	9-13	9-13#	10-5	10-5#	10-6	10-6#	10-7	10-7#	10-8	10-8#
	10-9	10-9#	10-10	10-10#	10-11	10-11#	10-12	10-12#	10-13	10-13#	10-14	10-14#
	10-15	10-15#	10-16	10-16#	10-17	10-17#	10-18	10-18#	10-19	10-19#	10-20	10-20#
	10-21	10-21#	10-22	10-22#	10-23	10-23#	10-24	10-24#	10-25	10-25#	10-26	10-26#
	10-27	10-27#	10-28	10-28#	10-29	10-29#	10-30	10-30#	10-31	10-31#	10-32	10-32#
	10-33	10-33#	10-34	10-34#	10-35	10-35#	10-36	10-36#	10-37	10-37#	10-38	10-38#

	10-39	10-39#	10-40	10-40#	10-41	10-41#	10-42	10-42#	10-43	10-43#	10-44	10-44#
	10-45	10-45#	10-46	10-46#	10-47	10-47#	10-48	10-48#	10-49	10-49#	10-50	10-50#
	10-51	10-51#	10-52	10-52#	11-5	11-5	11-5	11-5#	11-5#	11-6	11-6#	11-7
	11-7#	11-8	11-8#	11-9	11-9#	11-10	11-10#	11-11	11-11#	11-12	11-12#	11-13
	11-13#	11-14	11-14#	11-15	11-15#	11-16	11-16#	11-17	11-17#	11-18	11-18#	11-19
	11-19#	11-20	11-20#	11-21	11-21#	11-22	11-22#	11-23	11-23#	11-24	11-24#	11-25
	11-25#	11-26	11-26#	11-27	11-27#	11-28	11-28#	11-29	11-29#	11-30	11-30#	11-31
	11-31#	11-32	11-32#	11-33	11-33#	11-34	11-34#	11-35	11-35#	11-36	11-36#	11-37
	11-37#	11-38	11-38#	11-39	11-39#	11-40	11-40#	11-41	11-41#	12-5	12-5	12-5
	12-5#	12-5#	12-6	12-6#	12-7	12-7#	12-8	12-8#	12-9	12-9#	12-10	12-10#
	12-11	12-11#	12-12	12-12#	12-13	12-13#	13-4	13-4#	13-5	13-5#	13-6	13-6#
	13-7	13-7#	13-8	13-8#	13-9	13-9#	13-10	13-10#	13-11	13-11#	13-12	13-12#
	13-13	13-13#	13-14	13-14#	14-6	14-6#	14-7	14-7#	14-8	14-8#	14-9	14-9#
	14-10	14-10#	14-11	14-11#	14-12	14-12#	14-13	14-13#	14-14	14-14#	14-15	14-15#
	14-16	14-16#	14-17	14-17#	14-18	14-18#						
SVPAR5	8-14	8-14	15-24#	16-52*								
SVPAR6	8-15	8-15	15-25#	16-51*								
SVRO	8-6	8-6	15-17#									
SVR1	8-7	8-7	15-18#									
SVR2	8-8	8-8	15-19#									
SVR3	8-9	8-9	15-20#									
SVR4	8-10	8-10	15-21#									
SVR5	8-11	8-11	15-22#									
SVREG	15-23#	16-53										
SYSDAT	2-16	17-6										
SYTIMH	2-17	17-30										
SYTIML	2-17	17-31										
TACCR	32-8	34-8#										
TADR	15-57#	33-19*	33-30	34-5*	34-53*	35-29						
TATT	15-56#	33-18*	34-16*	34-26	34-41	34-52*	35-14	35-28				
TBUF	15-42#	16-46*	16-47*	39-18*								
TCLR	32-7	33-12#										
TCSR	15-43#	16-45*	39-16									
TECOL	15-53#	33-15*	33-31	34-15*	34-30	34-34	34-35*	34-40	34-45*	34-46	34-49*	
TEINX	15-55#	33-17*	34-23	34-24*	34-51*	35-22	35-43					
TENABL	8-5#	8-6	8-7	8-8	8-9	8-10	8-11	8-12	8-13	8-14	8-15	9-5#
	9-6	9-7	9-8	9-9	9-10	9-11	9-12	9-13	10-5#	10-6	10-7	10-8
	10-9	10-10	10-11	10-12	10-13	10-14	10-15	10-16	10-17	10-18	10-19	10-20
	10-21	10-22	10-23	10-24	10-25	10-26	10-27	10-28	10-29	10-30	10-31	10-32
	10-33	10-34	10-35	10-36	10-37	10-38	10-39	10-40	10-41	10-42	10-43	10-44
	10-45	10-46	10-47	10-48	10-49	10-50	10-51	10-52	11-5#	11-6	11-7	11-8
	11-9	11-10	11-11	11-12	11-13	11-14	11-15	11-16	11-17	11-18	11-19	11-20
	11-21	11-22	11-23	11-24	11-25	11-26	11-27	11-28	11-29	11-30	11-31	11-32
	11-33	11-34	11-35	11-36	11-37	11-38	11-39	11-40	11-41	12-5#	12-6	12-7
	12-8	12-9	12-10	12-11	12-12	12-13	13-4#	13-5	13-6	13-7	13-8	13-9
	13-10	13-11	13-12	13-13	13-14	14-6#	14-7	14-8	14-9	14-10	14-11	14-12
	14-13	14-14	14-15	14-16	14-17	14-18						
TINIT	32-6	33-4#										
TIOBAS	10-36	10-36	21-12									
TK1SEC	2-18	17-32										
TNHL	2-15	11-27										
TOTON	10-37	10-37										
TOUT	32-9	35-5#										
TRPAR5	2-13	16-52										
TSCOL	15-52#	33-14*	34-13*	34-29	34-43*	34-48*	35-40					
TSEMT	2-12	21-10										

TSEXEC	2-12	21-9					
TSGEN	2-12	21-7					
TSINIT	2-12	21-11					
TSINX	15-54#	33-16*	34-19	34-20*	34-50*	35-10	35-20
TSTIO	2-12	21-8					
TXARG	15-70#	19-7					
TXDEV	15-73#	19-20					
TXOID	15-72#	19-14					
TXSEG	15-71#						
UBUSMP	10-38	10-38					
UIOCNT	10-39	10-39					
URO	14-7	14-7					
USP	10-40	10-40					
USRBAS	10-41	10-41					
USRJOB	10-42	10-42					
VBUSTP	10-43	10-43					
VCSHNB	10-44	10-44					
VDMTCR	2-10	16-34					
VJCB	14-6#	29-28					
VMISC	10-5#	24-17					
VMXSF	10-45	10-45					
VMXSFC	10-46	10-46					
VNGR	10-47	10-47					
VNUMDC	10-48	10-48					
VPAR5	2-13	23-56					
VPAR6	2-13	23-58					
VPLAS	10-49	10-49					
VSWPFL	10-50	10-50					
VSWPSL	10-51	10-51					
VUXIFL	10-52	10-52					
XFLG	15-65#	39-11*	39-12*	39-13			

