

SSSSSSSSSSSS	DDDDDDDDDDDD	AAAAA
SSSSSSSSSSSS	DDDDDDDDDDDD	AAAAA
SSSSSSSSSSSS	DDDDDDDDDDDD	AAAAA
SSS	DDD	AAA
SSS	DDD	AAA
SSS	DDD	AAA
SSS	DDD	AAA
SSS	DDD	AAA
SSS	DDD	AAA
SSSSSSSSSS	DDD	AAA
SSSSSSSSSS	DDD	AAA
SSSSSSSSSS	DDD	AAA
SSS	DDD	AAAAA
SSS	DDD	AAAAA
SSS	DDD	AAAAA
SSS	DDD	AAA
SSS	DDD	AAA
SSS	DDD	AAA
SSSSSSSSSSSS	DDDDDDDDDDDD	AAA
SSSSSSSSSSSS	DDDDDDDDDDDD	AAA
SSSSSSSSSSSS	DDDDDDDDDDDD	AAA

```
SSSSSSSS TTTTTTTTTT AAAAAA CCCCCCCC KK KK SSSSSSSS
SSSSSSSS TTTTTTTTTT AAAAAA CCCCCCCC KK KK SSSSSSSS
SS      TT      AA      AA      CC      KK      KK      SS
SS      TT      AA      AA      CC      KK      KK      SS
SS      TT      AA      AA      CC      KK      KK      SS
SS      TT      AA      AA      CC      KK      KK      SS
SSSSSSS  TT      AA      AA      CC      KKKKKK  KK      SSSSSS
SSSSSSS  TT      AA      AA      CC      KKKKKK  KK      SSSSSS
SS      TT      AA      AA      CC      KK      KK      SS
SS      TT      AA      AA      CC      KK      KK      SS
SS      TT      AA      AA      CC      KK      KK      SS
SSSSSSSS TT      AA      AA      CCCCCCCC KK      KK      SSSSSSSS
SSSSSSSS TT      AA      AA      CCCCCCCC KK      KK      SSSSSSSS
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      IIIIII SSSSSSSS
LL      IIIIII SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```

STACKS
Table of contents

DUMP STACK MEMORY ROUTINES

K 2

16-SEP-1984 01:46:38 VAX/VMS Macro V04-00

Page 0

STA
V04

(1)	2	COPYRIGHT NOTICE
(1)	29	PROGRAM DESCRIPTION
(2)	69	DECLARATIONS
(3)	80	READ-ONLY DATA DEFINITIONS
(4)	102	PRINT_ANY_STACK, PRINT ANY RANGE AS STACK
(5)	161	PRINT_STACKS -- PRINT THE AVAILABLE STACKS
(6)	247	GET_POINTERS, OBTAIN STACK POINTERS
(7)	381	DUMP_STACK -- DUMP CONTENTS OF STACK

```
0000 1      .TITLE  STACKS DUMP STACK MEMORY ROUTINES
0000 2      .SBTTL  COPYRIGHT NOTICE
0000 3      .IDENT  'V04-000'
0000 4      :
0000 5      :*****
0000 6      :*
0000 7      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      :*  ALL RIGHTS RESERVED.
0000 10     :*
0000 11     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     :*  TRANSFERRED.
0000 17     :*
0000 18     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     :*  CORPORATION.
0000 21     :*
0000 22     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     :*
0000 25     :*
0000 26     :*****
0000 27     :
```

```
0000 29 .SBTTL PROGRAM DESCRIPTION
0000 30 :++
0000 31 FACILITY
0000 32 :
0000 33 SYSTEM DUMP ANALYZER
0000 34 :
0000 35 ABSTRACT
0000 36 :
0000 37 DUMP STACK MEMORY ROUTINES
0000 38 :
0000 39 ENVIRONMENT
0000 40 :
0000 41 NATIVE MODE, USER MODE
0000 42 :
0000 43 AUTHOR
0000 44 :
0000 45 TIM HALVORSEN, JULY 1978
0000 46 :
0000 47 MODIFIED BY
0000 48 :
0000 49 V005 TMH0005 Tim Halvorsen 19-Oct-1981
0000 50 Fix KTA0023 to correctly read stack limit/base arrays
0000 51 (it wasn't doing an indirection through the required
0000 52 symbol pointers to get the array address in P1 space).
0000 53 :
0000 54 V03-004 KTA0023 Kerbey Altmann 05-Jul-1981
0000 55 Change method of calculating stack sizes so as
0000 56 to conform to new array of limits.
0000 57 :
0000 58 V003 MTR0001 Mike Rhodes 22-Jun-1981
0000 59 Change all CMPW's referencing an MSG$_ symbol to CMPL's.
0000 60 Remove references to SSDAMSGDEF macro.
0000 61 :
0000 62 V002 TMH0002 Tim Halvorsen 20-May-1981
0000 63 Add routine to print any range of memory as a stack.
0000 64 :
0000 65 V001 TMH0001 Tim Halvorsen 07-Feb-1981
0000 66 Change word displacements to longword displacements.
0000 67 :--
```

0000	69	.SBTTL	DECLARATIONS	
0000	70	:		
0000	71	:		
0000	72	:	SYMBOL DEFINITIONS	
0000	73	:		
0000	74		\$VADEF	: VIRTUAL ADDRESS DEFINITIONS
0000	75		\$EMBDEF <CR>	: CRASHDUMP ERROR LOG ENTRY
0000	76		\$PCBDEF	: PROCESS CONTROL BLOCK
0000	77		\$PHDDEF	: PROCESS HEADER BLOCK
0000	78		\$PSLDEF	: PROCESS STATUS LONGWORD
0000			\$OPTDEF	: DEFINE OPTION BITS

```
0000 80 .SBTTL READ-ONLY DATA DEFINITIONS
0000 81
0000 82 :
0000 83 : READ-ONLY DATA DEFINITIONS
0000 84 :
0000 85
00000000 86 .PSECT STACKS,EXE,NOWRT
0000 87
0000 88 .DEFAULT DISPLACEMENT, LONG
0000 89
0000 90 INTERRUPT:
54 50 55 52 52 45 54 4E 49 00' 0000 91 .ASCIC 'INTERRUPT'
09 0000
000A 92 MODES:
4C 45 4E 52 45 4B 00' 000A 93 .ASCIC 'KERNEL'
06 000A
45 56 49 54 55 43 45 58 45 00' 0011 94 .ASCIC 'EXECUTIVE'
09 0011
52 4F 53 49 56 52 45 50 55 53 00' 001B 95 .ASCIC 'SUPERVISOR'
0A 001B
52 45 53 55 00' 0026 96 .ASCIC 'USER'
04 0026
002B 97 SWAPPER:
52 45 4B 20 52 45 50 50 41 57 53 00' 002B 98 .ASCIC 'SWAPPER KERNEL'
4C 45 4E 0037
0E 002B
003A
4E 57 4F 4E 4B 4E 55 00' 003A 99 UNKNOWN:
07 003A 100 .ASCIC 'UNKNOWN'
```

```
0042 102 .SBTTL PRINT_ANY_STACK, PRINT ANY RANGE AS STACK
0042 103 :---
0042 104 :
0042 105 PRINT ANY RANGE OF MEMORY AS A STACK, ONE LONGWORD PER LINE,
0042 106 WITH SYMBOLIZATION OF THE LONGWORDS.
0042 107 :
0042 108 INPUTS:
0042 109 :
0042 110 OPTIONS = OPTIONS FLAGS (RANGE OR LENGTH BITS RELEVANT)
0042 111 ESP = BASE OF STACK (HIGHEST ADDRESS)
0042 112 (OR, IF LENGTH BIT SET)
0042 113 ESP = SIZE OF STACK
0042 114 ESP+4 = LOWER LIMIT OF STACK
0042 115 :
0042 116 OUTPUTS:
0042 117 :
0042 118 NONE
0042 119 :---
0042 120 .ENABL LSB
0042 121 :
001C 0042 122 .ENTRY PRINT_ANY_STACK, ^M<R2,R3,R4>
0044 123 :
52 00000000'EF D0 0044 124 MOVL OPTIONS, R2
51 00000000'EF D0 004B 125 MOVL ESP, R1 ; POINT TO EXPRESSION STACK
07 52 03 E0 0052 126 BBS #OPT$V_RANGE, R2, 10$ ; RANGE SPECIFIED
0C 52 04 E0 0056 127 BBS #OPT$V_LENGTH, R2, 20$ ; LENGTH SPECIFIED
50 D4 005A 128 CLRL R0 ; SYNTAX ERROR
04 005C 129 RET
005D 130 :
52 04 A1 D0 005D 131 10$: MOVL 4(R1), R2 ; R2 = LOWEST ADDRESS (LIMIT)
53 61 D0 0061 132 MOVL (R1), R3 ; R3 = HIGHEST ADDRESS (BASE)
09 11 0064 133 BRB 30$
0066 134 :
53 52 04 A1 D0 0066 135 20$: MOVL 4(R1), R2 ; R2 = LOWEST ADDRESS (LIMIT)
04 A1 61 C1 006A 136 ADDL3 (R1), 4(R1), R3 ; R3 = HIGHEST ADDRESS (BASE)
53 52 D1 006F 137 :
4A 1E 0072 138 30$: CMPL R2, R3 ; CHECK IF DONE YET
0074 139 BGEQU 90$ ; BRANCH IF NOT
007D 140 TRYMEM (R2) ; GET NEXT LONGWORD
31 50 E9 007D 141 BLBC R0, 80$ ; BRANCH IF NOT FOUND
54 51 D0 0080 142 MOVL R1, R4 ; SAVE IN R3
0083 143 ALLOC 40, -(SP) ; RESULT BUFFER
008D 144 PUSHL R4
00000000'EF 02 FB 008F 145 CALLS #2, SYMBOLIZE ; ATTEMPT TO SYMBOLIZE
51 DD 0096 146 PUSHL R1 ; SYMBOL STRING
54 DD 0098 147 PUSHL R4 ; CONTENTS OF LONGWORD
52 DD 009A 148 PUSHL R2 ; ADDRESS OF LONGWORD
009C 149 PRINT 3, <!! !XL !XL!_!AS>
5E 30 C0 00A9 150 ADDL #40+8, SP ; DEALLOCATE BUFFER
52 04 C0 00AC 151 ADDL2 #4, R2 ; NEXT LONGWORD
BE 11 00AF 152 BRB 30$ ; LOOP UNTIL DONE
00B1 153 :
00B1 154 80$: PRINT 0, <!!_!(Stack not in physical memory)>
00BE 155 :
50 01 D0 00BE 156 90$: MOVL #1, R0 ; SUCCEED
04 00C1 157 RET
00C2 158 :
```

STACKS
V04-000

D 3
DUMP STACK MEMORY ROUTINES
PRINT_ANY_STACK, PRINT ANY RANGE AS STAC
00C2 159 .DSABL LSB

16-SEP-1984 01:46:38 VAX/VMS Macro V04-00
5-SEP-1984 03:34:31 [SDA.SRC]STACKS.MAR;1

Page 6
(4)

SYN
V04

```
00C2 161 .SBTTL PRINT_STACKS -- PRINT THE AVAILABLE STACKS
00C2 162 :---
00C2 163 :
00C2 164 PRINT_STACKS
00C2 165 :
00C2 166 THIS ROUTINE DETERMINES WHICH STACK IS CURRENTLY BEING
00C2 167 RUN AND PRINTS THAT STACK. IN ADDITION, IT ALSO PRINTS
00C2 168 ALL THE REMAINING PROCESS STACKS TO FOLLOW CALL CHAINS.
00C2 169 :
00C2 170 INPUTS:
00C2 171 :
00C2 172 ERLPTR = ADDRESS OF ERROR LOG ENTRY
00C2 173 :
00C2 174 OUTPUTS:
00C2 175 :
00C2 176 NONE
00C2 177 :
00C2 178 :---
00C2 179 :
00C2 180 .ENABL LSB
00C2 181 :
07FC 00C2 182 PRINT_STACKS::
00C2 183 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
00C4 184 :
00C4 185 :
00C4 186 SETUP ARRAY CONTAINING EACH STACK POINTER, BASE AND LENGTH
00C4 187 OF STACK, AND ADDRESS OF TEXT DESCRIBING STACK.
00C4 188 :
00C1 30 00C4 189 BSBW GET_POINTERS ; SETUP STACK ARRAY
00C7 190 R8 = ADDRESS OF 5-ENTRY 4-LONGWORD ARRAY
00C7 191 R9 = ADDRESS OF ARRAY ENTRY FOR STACK CURRENTLY IN USE
00C7 192 :
00C7 193 PRINT THE CURRENT OPERATING STACK IF NO OPTIONS SELECTED
00C7 194 :
56 00000000'EF D0 00C7 195 MOVL OPTIONS,R6 ; BIT MASK OF STACKS TO DUMP
3D 12 00CE 196 BNEQ 50$ ; IF WANTS GIVEN STACK TYPE
00D0 197 SUBHD <Current operating stack>
59 D5 00DD 198 TSTL R9 ; IS STACK POINTER KNOWN?
OF 13 00DF 199 BEQL 20$ ; BRANCH IF NOT
50 50 59 58 C3 00E1 200 SUBL3 R8,R9,R0 ; GET OFFSET INTO ARRAY
50 50 FC 8F 78 00E5 201 ASHL #-4,R0,R0 ; CALCULATE INDEX INTO ARRAY
56 01 50 78 00EA 202 ASHL R0,#1,R6 ; SET TO PRINT CURRENT STACK
2A 11 00EE 203 BRB 55$
00F0 204 20$:
56 01 D0 00F0 205 MOVL #1,R6 ; SET TO PRINT 1 STACK
59 58 D0 00F3 206 MOVL R8,R9 ; USE INTERRUPT STACK SLOT
69 5A D0 00F6 207 MOVL R10,(R9) ; SET STACK POINTER
04 A9 FE00 CA 9E 00F9 208 MOVAB -512(R10),4(R9) ; SET BASE 1 PAGE AWAY
08 A9 0200 8F 3C 00FF 209 MOVZWL #512,8(R9) ; SET LENGTH TO 1 PAGE
0C A9 FF31 CF 9E 0105 210 MOVAB UNKNOWN,12(R9) ; SET TEXT STRING ADDRESS
OD 11 010B 211 BRB 55$
010D 212 :
010D 213 :
010D 214 :
010D 215 50$:
010D 216 :
011A 217 55$: SUBHD <Process stacks> ; SET PAGE HEADING
```

```
57 01 D0 011A 218 SKIP PAGE
0121 219 MOVL #^B1,R7 ; START AT INTERRUPT STACK
0124 220 60$: BITL R7,R6 ; CHECK IF REQUESTED
56 57 D3 0124 221 BEQL 70$ ; BRANCH IF NOT
4B 13 0127 222 BICL2 R7,R6 ; INDICATE PRINTED
56 57 CA 0129 223 TSTL 8(R8) ; CHECK LENGTH OF STACK
08 A8 D5 012C 224 BEQL 70$ ; SKIP IF COULD NOT LOCATE
43 13 012F 225 PUSHL (R8) ; PUSH STACK POINTER
68 DD 0131 226 PUSHL 8(R8) ; LENGTH OF STACK SPACE
08 A8 DD 0133 227 PUSHL 4(R8) ; BASE STACK ADDRESS
04 A8 DD 0136 228 PUSHL 12(R8) ; ADDRESS OF TEXT STRING
0C A8 DD 0139 229 SKIP PAGE
013C 230 59 58 D1 0143 231 CMPL R8,R9 ; IS THIS CURRENT STACK?
OF 12 0146 232 BNEQ 65$ ; BRANCH IF NOT
0148 233 PRINT 1,<Current operating stack (!AC):>
0D 11 0155 234 BRB 68$
0157 235 65$: PRINT 1,<!AC stack:>
0164 236 68$: SKIP 1
0000032A'EF 03 FB 016D 237 CALLS #3,DUMP_STACK ; DUMP THE STACK
57 57 01 78 0174 238 70$: ASHL #1,R7,R7 ; SHIFT BIT TO NEXT STACK
58 10 C0 0178 239 ADDL #16,R8 ; SKIP TO NEXT ENTRY
10 57 D1 017B 240 CMPL R7,#^B10000 ; CHECK IF DONE
A4 15 017E 241 BLEQ 60$ ; CONTINUE UNTIL DONE
0180 242 STATUS SUCCESS
04 0187 243 RET
0188 244
0188 245 .DSABL LSB
```

```
0188 247 .SBTTL GET_POINTERS, OBTAIN STACK POINTERS
0188 248 :---
0188 249 :
0188 250 : THIS SUBROUTINE IS USED TO OBTAIN THE STACK POINTERS
0188 251 : FOR THE CURRENTLY SELECTED PROCESS. IT RETURNS AN ARRAY
0188 252 : OF ENTRIES FOR EACH OF THE POSSIBLE STACKS (ISP THRU USP).
0188 253 : EACH ENTRY CONTAINS THE STACK POINTER, STACK BASE, LENGTH
0188 254 : AND ASCII STRING DESCRIBING ENTRY FOR DISPLAY PURPOSES.
0188 255 :
0188 256 : CALLING SEQUENCE:
0188 257 :
0188 258 : ERLPTR = ADDRESS OF ERROR LOG ENTRY
0188 259 : PROC_INDEX = INDEX OF CURRENTLY SELECTED PROCESS
0188 260 : JSB
0188 261 : R8 = ADDRESS OF ARRAY (5 4-LONGWORD ENTRIES)
0188 262 :     ISP     BASE     SIZE     TEXT
0188 263 :     KSP     BASE     SIZE     TEXT
0188 264 :     ESP     BASE     SIZE     TEXT
0188 265 :     SSP     BASE     SIZE     TEXT
0188 266 :     USP     BASE     (INFINITE) TEXT
0188 267 : R9 = ADDRESS OF ARRAY ENTRY FOR STACK CURRENTLY IN USE
0188 268 :     IF CURRENTLY USING UNKNOWN STACK, R9 = 0
0188 269 : R10 = CURRENT STACK POINTER
0188 270 :
0188 271 :---
0188 272 :
0188 273 GET_POINTERS:
0188 274 :
0188 275 : ALLOCATE ARRAY ON CALLER'S STACK
0188 276 :
0188 277 : POPL      R0                ; SAVE RETURN ADDRESS
0188 278 : SUBL      #5*4*4,SP        ; 5 4-LONGWORD ENTRIES
0188 279 : MOVL      SP,R8            ; ADDRESS OF ARRAY
0188 280 : PUSHL     R0               ; SET RETURN ADDRESS
0188 281 : PUSHR     #^M<R2,R3,R4,R5> ; SAVE REGISTERS
0188 282 : MOVC5     #0,(SP),#0,#5*4*4,(R8) ; PRE-ZERO ARRAY
0188 283 : PUSHL     R8               ; SAVE ADDRESS OF ARRAY
0188 284 :
0188 285 : FILL ENTRY FOR INTERRUPT STACK
0188 286 :
0188 287 : MOVL      ERLPTR,R5        ; ADDRESS OF ERROR LOG ENTRY
0188 288 : MOVL      EMB$LCR ISP(R5),(R8)+ ; SETUP ISP ENTRY
0188 289 : REQMEM    @EXE$GL RPB      ; RESTART PARAMETER BLOCK
0188 290 : SUBL3     R1,-4(R8),R0     ; IS ISP IN RPB PAGE?
0188 291 : BLSS     5$                ; BRANCH IF NOT
0188 292 : CMPL     R0,#512           ; RPB IS ONLY 1 PAGE
0188 293 : BGTR     5$                ; BRANCH IF NOT IN RPB
0188 294 : ADDL3     R1,#512,(R8)+    ; BASE ADDRESS OF RPB STACK
0188 295 : MOVL      #1,R1           ; SIZE OF STACK IN PAGES
0188 296 : BRB      8$                ; SET STACK SIZE
0188 297 5$: REQMEM    @EXE$GL_INTSTK,(R8)+ ; BASE OF INTERRUPT STACK
0188 298 : REQMEM    @SGN$GW_ISPPGCT ; INTERRUPT STACK PAGE COUNT
0188 299 : MOVZWL    R1,R1           ; ZERO FILL TO LONGWORD
0188 300 8$: ASHL     #9,R1,(R8)+ ; SET INTERRUPT STACK SIZE
0188 301 : MOVAB     INTERRUPT,(R8)+ ; ADDRESS OF TEXT STRING
0188 302 :
0188 303 : GET CURRENT SP AND ADDRESS OF PROCESS STACK POINTERS
```

5E	00000050	8F	C2	0188	277
	58	5E	D0	0192	279
		50	DD	0195	280
		3C	BB	0197	281
68	0050 8F	00	6E	00	2C
		58	DD	01A1	283
				01A3	284
				01A3	285
				01A3	286
55	00000000	'EF	D0	01A3	287
	88	20 A5	D0	01AA	288
				01AE	289
50	FC A8	51	C3	01BB	290
		16	19	01C0	291
00000200	8F	50	D1	01C2	292
		0D	14	01C9	293
88	00000200	8F	51	01CB	294
		51	01	01D3	295
		20	11	01D6	296
				01D8	297
				01E8	298
	51	51	3C	01F5	299
88	51	09	78	01F8	300
88	FE00	CF	9E	01FC	301
				0201	302
				0201	303

```
52 00000000'EF 3C 0201 304 ;
016C C5 52 B1 0208 305 ; MOVZWL PROC_INDEX,R2 ; GET CURRENTLY SELECTED INDEX
OA 12 020D 306 ; CMPW R2,EMB$C_CR_LENGTH+PCB$C_PID(R5) ; CRASH PROCESS?
52 10 A5 9E 020F 307 ; BNEQ 10$ ; BRANCH IF SOME OTHER PROCESS
53 5C A5 D0 0213 308 ; MOVAB EMB$C_CR_KSP(R5),R2 ; R2 = ADDRESS OF KSP-USP
58 11 0217 309 ; MOVL EMB$C_CR_SP(R5),R3 ; R3 = STACK POINTER
51 6142 DE 0219 310 ; BRB 20$
0219 311 10$ : REQMEM @SCH$GL_PCBVEC ; VECTOR OF PCB ADDRESSES
0226 312 ; MOVAL (R1)[R2],R1 ; ADDRESS OF POINTER TO PCB
022A 313 ; REQMEM (R1) ; GET ADDRESS OF PCB
0233 314 ; REQMEM PCB$C_PHD(R1) ; GET ADDRESS OF PHD
023D 315 ; REQMEM (R1),PHD,#PHD$C_LENGTH ; READ ENTIRE PROCESS HEADER
52 00000078'EF 9E 0252 316 ; MOVAB PHD+PHD$C_KSP,R2 ; R2 = ADDRESS OF KSP-USP
50 000000C4'EF D0 0259 317 ; MOVL PHD+PHD$C_PSL,R0 ; GET PROCESS STATUS LONGWORD
53 20 A5 D0 0260 318 ; MOVL EMB$C_CR_TSP(R5),R3 ; ASSUME ON INTERRUPT STACK
09 50 1A E0 0264 319 ; BBS #PSL$V_IS,R0,20$ ; BRANCH IF OK
50 50 02 18 EF 0268 320 ; EXTZV #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0 ; GET CURRENT MODE
53 6240 D0 026D 321 ; MOVL (R2)[R0],R3 ; GET PROCESS STACK POINTER
0271 322 :
0271 323 :
0271 324 :
54 5E 20 C2 0271 325 20$ : SUBL #8*4,SP ; ALLOCATE 8 LONGWORD ARRAY
10 AE DE 0274 326 ; MOVAL 4*4(SP),R4 ; R4 = LIMIT ADDRESS BUFFER
0278 327 ; TRYMEM @CTL$AL_STACKLIM,(R4),#4*4 ; READ 4 LIMIT ADDRESSES
3E 50 E9 0289 328 ; BLBC R0,45$ ; BRANCH IF CANNOT READ
54 5E D0 028C 329 ; MOVL SP,R4 ; R4 = BASE ADDRESS BUFFER
028F 330 ; TRYMEM @CTL$AL_STACK,(R4),#4*4 ; READ 4 BASE ADDRESSES
27 50 E9 02A0 331 ; BLBC R0,45$ ; BRANCH IF CANNOT READ
55 FD63 CF 9E 02A3 332 ; MOVAB MODES,R5 ; ADDRESS OF TEXT STRINGS
50 04 D0 02A8 333 ; MOVL #4,R0 ; ITERATION COUNT
02AB 334 30$ :
88 82 D0 02AB 335 ; MOVL (R2)+,(R8)+ ; SET STACK POINTER
88 64 D0 02AE 336 ; MOVL (R4),(R8)+ ; SET BASE OF STACK
84 10 A4 C3 02B1 337 ; SUBL3 4*4(R4),(R4)+,(R8)+ ; SET LENGTH OF STACK SPACE
06 14 02B6 338 ; BGTR 40$ ; BRANCH IF OK
FC A8 FFFF 8F 3C 02B8 339 ; MOVZWL #XFFFF,-4(R8) ; USER STACK HAS INFINITE LENGTH
88 55 D0 02BE 340 40$ : MOVL R5,(R8)+ ; ADDRESS OF TEXT STRING
51 85 9A 02C1 341 ; MOVZBL (R5)+,R1 ; LENGTH OF STRING
55 51 C0 02C4 342 ; ADDL R1,R5 ; SKIP TO NEXT STRING
E1 50 F5 02C7 343 ; SOBGTR R0,30$ ; CONTINUE UNTIL DONE
02CA 344 45$ :
5E 20 C0 02CA 345 ; ADDL #8*4,SP ; DEALLOCATE BUFFER SPACE
02CD 346 :
02CD 347 :
02CD 348 :
58 6E D0 02CD 349 ; MOVL (SP),R8 ; RESET R8 TO START OF ARRAY
50 05 D0 02D0 350 ; MOVL #5,R0 ; ITERATION COUNT
02D3 351 50$ :
04 A8 53 D1 02D3 352 ; CMPL R3,4(R8) ; CHECK IF BELOW BASE ADDRESS
10 1E 02D7 353 ; BGEQU 60$ ; BRANCH IF NOT
51 04 A8 08 A8 C3 02D9 354 ; SUBL3 8(R8),4(R8),R1 ; CALCULATE LOWEST ADDRESS
51 53 D1 02DF 355 ; CMPL R3,R1 ; CHECK IF ABOVE LOW ADDRESS
05 1F 02E2 356 ; BLSSU 60$ ; BRANCH IF NOT
59 58 D0 02E4 357 ; MOVL R8,R9 ; SET ADDRESS OF CURRENT ENTRY
38 11 02E7 358 ; BRB 90$ ; AND EXIT
58 10 C0 02E9 359 60$ : ADDL #16,R8 ; SKIP TO NEXT ENTRY
E4 50 F5 02EC 360 ; SOBGTR R0,50$ ; CONTINUE UNTIL ALL CHECKED
```

```

                                02EF 361 :
                                02EF 362 :
                                02EF 363 :
                                02EF 364 :
50 00000000'EF 59 D4 02EF 365 :
                                02F1 366 :
                                02F9 367 :
00000000'EF 50 D1 02FB 368 :
                                0302 369 :
                                0304 370 :
                                0308 371 :
                                030B 372 :
88 00000000'EF D0 030E 373 :
88 00000000'EF D0 0315 374 :
88 FDOB CF 9E 031C 375 :
                                0321 376 90$:
                                0324 377 :
                                0327 378 :
                                0329 379 :

CLRL R9 ; ASSUME UNKNOWN ADDRESS
SUBL3 R3,SWP$A_KSTK,R0 ; CHECK IF BELOW BASE ADDRESS
BLSS 90$ ; BRANCH IF NOT
CMPL R0,SWP$K_KSTKSZ ; CHECK IF WITHIN SWAPPER STACK
BGTRU 90$ ; BRANCH IF NOT IN RANGE
ADDL3 #16,(SP),R9 ; SET TO KERNEL STACK
MOVL R9,R8
MOVL R3,(R8)+ ; SET STACK POINTER
MOVL SWP$A_KSTK,(R8)+ ; SET SWAPPER BASE ADDRESS
MOVL SWP$K_KSTKSZ,(R8)+ ; SET SWAPPER STACK SIZE
MOVAB SWAPPER,(R8)+ ; SET SWAPPER TEXT STRING
POPL R8 ; RESTORE ADDRESS OF ARRAY
MOVL R3,R10 ; RETURN CURRENT STACK POINTER
POPR #^M<R2,R3,R4,R5> ; RESTORE REGISTERS
RSB
```

```
032A 381 .SBTTL DUMP_STACK -- DUMP CONTENTS OF STACK
032A 382 ---
032A 383
032A 384 DUMP_STACK
032A 385
032A 386 THIS ROUTINE PRINTS THE CONTENTS OF THE SPECIFIED
032A 387 STACK, 1 LONGWORD PER LINE. AN ATTEMPT IS MADE
032A 388 TO PRINT THE SYMBOLIC VALUE OF EACH LONGWORD NEXT
032A 389 TO ITS HEXIDECIMAL VALUE. SOME OF THE MEMORY PRECEEDING
032A 390 THE STACK POINTER IS DUMPED AS IT IS SOMETIMES
032A 391 USEFUL IN DEBUGGING.
032A 392
032A 393 INPUTS:
032A 394
032A 395 4(AP) = INITIAL ADDRESS OF STACK
032A 396 8(AP) = LENGTH OF STACK SPACE
032A 397 12(AP) = STACK POINTER
032A 398
032A 399 OUTPUTS:
032A 400
032A 401 NONE
032A 402
032A 403 ---
032A 404
00000600 032A 405 STACKLIM = 3*512 ; MAX. BYTES TO DUMP
032A 406
032A 407 .ENABL LSB
032A 408
000C 032A 409 DUMP_STACK::
032A 410 .WORD ^M<R2,R3>
032C 411
032C 412 SUBL3 #32,12(AP),R2 ; 32 BYTES PRECEEDING SP
51 52 0C AC 20 C3 0331 413 SUBL3 12(AP),4(AP),R1 ; STACK SPACE IN USE
04 AC 0C AC 0C AC C3 0337 414 BLSSU 1$ ; IF SP OUT OF BOUNDS
08 AC 51 D1 0339 415 CMPL R1,8(AP) ; CHECK IF BEYOND STACK
09 1A 033D 416 BGTRU 1$ ; IF SP OUT OF BOUNDS
00000600 8F 51 D1 033F 417 CMPL R1,#STACKLIM ; MAX. DUMP SIZE
0C 15 0346 418 BLEQ 5$ ; BRANCH IF OK
04 AC 0C AC 00000600 8F C1 0348 419 1$: ADDL3 #STACKLIM,12(AP),4(AP) ; SET NEW ENDING ADDRESS
0E 11 0352 420 BRB 10$ ; NEVER RESET STARTING ADDR.
0354 421 5$:
51 04 AC 08 AC C3 0354 422 SUBL3 8(AP),4(AP),R1 ; COMPUTE BASE ADDRESS
51 52 D1 035A 423 CMPL R2,R1 ; UNLESS IT RUNS OVER
03 1E 035D 424 BGEQU 10$ ; IF NOT, GO AHEAD
52 51 D0 035F 425 MOVL R1,R2 ; IF SO, START AT THE BASE
0362 426 10$:
04 AC 52 D1 0362 427 CMPL R2,4(AP) ; CHECK IF DONE YET
03 1F 0366 428 BLSSU 15$ ; BRANCH IF NOT
006A 31 0368 429 BRW 90$ ; EXIT - DONE
036B 430 15$:
036B 431 TRYMEM (R2) ; GET NEXT LONGWORD
50 50 E9 0374 432 BLBC R0,80$ ; BRANCH IF NOT FOUND
53 51 D0 0377 433 MOVL R1,R3 ; SAVE IN R3
037A 434 ALLOC 40,-(SP) ; RESULT BUFFER
53 DD 0384 435 PUSHL R3
00000000'EF 02 FB 0386 436 CALLS #2,SYMBOLIZE ; ATTEMPT TO SYMBOLIZE
51 DD 038D 437 PUSHL R1 ; SYMBOL STRING
```

```

OC AC 53 DD 038F 438 PUSHL R3 ; CONTENTS OF LONGWORD
52 DD 0391 439 PUSHL R2 ; ADDRESS OF LONGWORD
52 D1 0393 440 CMPL R2,12(AP) ; IS THIS TOP OF STACK?
18 12 0397 441 BNEQ 20$ ; BRANCH IF NOT
0399 442 SKIP 1
03A2 443 PRINT 3,^/!_ SP => !XL !XL!_!AS/
OD 11 03AF 444 BRB 50$
03B1 445 20$: PRINT 3,<!_!_!XL !XL!_!AS>
03B1 446 50$: ADDL #40+8,SP ; DEALLOCATE BUFFER
SE 30 C0 03BE 448 ADDL2 #4,R2 ; NEXT LONGWORD
52 04 C0 03C1 449 BRW 10$ ; LOOP UNTIL DONE
FF9B 31 03C4 450 80$: PRINT 0,<!_!_(Stack not in physical memory)>
03C7 451 RET
03C7 452
04 03D4 453 90$: CMPL R2,12(AP) ; ARE WE AT TOP OF STACK?
OC AC 52 D1 03D5 455 BNEQ 95$ ; BRANCH IF NOT
16 12 03D9 456 SKIP 1
03DB 457 PRINT 0,^/!_ SP => (STACK IS EMPTY)/
03E4 458 95$: RET
04 03F1 459 .DSABL LSB
03F2 461
03F2 462
```

STACKS
V04-000

DUMP STACK MEMORY ROUTINES
DUMP_STACK -- DUMP CONTENTS OF STACK

L 3

16-SEP-1984 01:46:38 VAX/VMS Macro V04-00
5-SEP-1984 03:34:31 [SDA.SRC]STACKS.MAR;1

Page 14
(9)

03F2 464
03F2 465 .END

SYM
V04

54

51
54
4F

44
4E

44

4C

49

4E

46

45

43

54

58

43

46

50

44

52

54

STACKS
Symbol table

DUMP STACK MEMORY ROUTINES

M 3

16-SEP-1984 01:46:38
5-SEP-1984 03:34:31VAX/VMS Macro V04-00
[SDA.SRC]STACKS.MAR;1Page 15
(9)

ARGS	= 00000001		
CTLSAL_STACK	*****	X	02
CTLSAL_STACKLIM	*****	X	02
DUMP_STACK	0000032A	RG	02
EMBSL_CR_LENGTH	= 0000010C		
EMBSL_CR_ISP	= 00000020		
EMBSL_CR_KSP	= 00000010		
EMBSL_CR_SP	= 0000005C		
ERLPT	*****	X	02
ESP	*****	X	02
EXESGL_INTSTK	*****	X	02
EXESGL_RPB	*****	X	02
GET_POINTERS	00000188	R	02
INTERRUPT	00000000	R	02
MODES	0000000A	R	02
MSG\$ SUCCESS	*****	X	02
NEW_PAGE	*****	X	02
OPTSV_LENGTH	= 00000004		
OPTSV_RANGE	= 00000003		
OPTIONS	*****	X	02
PCBSL_PHD	= 0000006C		
PCBSL_PID	= 00000060		
PHD	*****	X	02
PHD\$C_LENGTH	= 0000017C		
PHD\$C_KSP	= 00000078		
PHD\$C_PSL	= 000000C4		
PRINT	*****	X	02
PRINT_ANY_STACK	00000042	RG	02
PRINT_STACKS	000000C2	RG	02
PROC_INDEX	*****	X	02
PSL\$S_CURMOD	= 00000002		
PSL\$V_CURMOD	= 00000018		
PSL\$V_IS	= 0000001A		
REQMEM	*****	X	02
SCH\$GL_PCBVEC	*****	X	02
SET_HEADING	*****	X	02
SGN\$GW_ISPPGCT	*****	X	02
SKIP_LINES	*****	X	02
STACKLIM	= 00000600		
SWAPPER	0000002B	R	02
SWPSA_KSTK	*****	X	02
SWPSK_KSTKSZ	*****	X	02
SYMBOLIZE	*****	X	02
TRYMEM	*****	X	02
UNKNOWN	0000003A	R	02

+-----+
! Psect synopsis !
+-----+

PSECT name

Allocation

PSECT No.

Attributes

ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
STACKS	000003F2 (1010.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE
LITERALS	00000134 (308.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	35	00:00:00.05	00:00:02.00
Command processing	133	00:00:00.45	00:00:01.86
Pass 1	257	00:00:04.82	00:00:18.79
Symbol table sort	0	00:00:00.64	00:00:02.30
Pass 2	96	00:00:01.20	00:00:04.17
Symbol table output	6	00:00:00.05	00:00:00.56
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	531	00:00:07.23	00:00:29.70

The working set limit was 1500 pages.

42929 bytes (84 pages) of virtual memory were used to buffer the intermediate code.

There were 40 pages of symbol table space allocated to hold 622 non-local and 41 local symbols.

465 source lines were read in Pass 1, producing 22 object records in Pass 2.

24 pages of virtual memory were used to define 23 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SDA.OBJ]SDALIB.MLB;1	9
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	7
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	20

763 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:STACKS/OBJ=OBJ\$:STACKS MSRC\$:STACKS/UPDATE=(ENH\$:STACKS)+EXECML\$/LIB+LIB\$:SDALIB/LIB

0354 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

SMGRTL

SMGBLDTM
MAP

SDAMSG
LIS

VAXINST
LIS

SMGMAPTRM
MAP

SMGKCB
SDL

VALIDATE
LIS

STACKS
LIS

SMGDEF
SDL

SMGKDE
SDL

SMGSHR
MAP