


```

MM      MM      AAAAAA  PPPPPPPP  PPPPPPPP  IIIIII  NN      NN      GGGGGGGG
MM      MM      AAAAAA  PPPPPPPP  PPPPPPPP  IIIIII  NN      NN      GGGGGGGG
MMM     MMM     AA       AA     PP       PP     PP       PP     II      NN      NN      GG      GGGGGGGG
MMM     MMM     AA       AA     PP       PP     PP       PP     II      NN      NN      GG      GGGGGGGG
MM      MM      AA       AA     PP       PP     PP       PP     II      NNNN   NN      GG      GGGGGGGG
MM      MM      AA       AA     PP       PP     PP       PP     II      NNNN   NN      GG      GGGGGGGG
MM      MM      AA       AA     PPPPPPPP  PPPPPPPP  II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AA       AA     PPPPPPPP  PPPPPPPP  II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AAAAAAAAAA  PP       PP     II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AAAAAAAAAA  PP       PP     II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AA       AA     PP       PP     II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AA       AA     PP       PP     II      NN      NN      NN      GG      GGGGGGGG
MM      MM      AA       AA     PP       PP     IIIIII  NN      NN      NN      GGGGGGGG
MM      MM      AA       AA     PP       PP     IIIIII  NN      NN      NN      GGGGGGGG

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
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      IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS

```

(1)	2	COPYRIGHT NOTICE
(2)	29	PROGRAM DESCRIPTION
(3)	99	DECLARATIONS
(4)	113	STORAGE DEFINITIONS
(5)	147	MAP_DUMP - MAP THE DUMP INTO VIRTUAL MEMORY
(6)	252	SAVE_DUMP, Save dump file into another file
(7)	324	MARK_DUMP -- MARK DUMP ANALYZED
(8)	380	GETMEM - READ DUMP MEMORY AREA
(9)	484	PUTMEM, STORE INTO MAPPED MEMORY RANGE
(10)	540	MAPMEM, MAP A GIVEN ADDRESS RANGE INTO LOCAL MEMORY
(11)	625	LOCATE_PFN, FIND PAGE WITHIN DUMP FILE

```
0000 1 .TITLE MAPPING DUMP MEMORY MAPPING 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 MEMORY MAPPING 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 V03-005 MSH0070 Michael S. Harvey 24-Jul-1984
0000 50 Close output file if an error occurs while writing to it.
0000 51
0000 52 V03-004 EMB0103 Ellen M. Batbouta 11-Jun-1984
0000 53 Remove check for a dump file size less than 32 meg
0000 54 in routine, MAP_DUMP. This check is no longer nec-
0000 55 cessary and prevents analyzing dump of this size or
0000 56 larger.
0000 57
0000 58 V03-003 EMD0081 Ellen M. Dusseault 11-Apr-1984
0000 59 Display warning message, SDA-W-NOTCOPIED, if the copy
0000 60 command is issued while analyzing the current system.
0000 61
0000 62 V03-002 LMP0028 L. Mark Pilant, 10-Jun-1982 14:35
0000 63 Adjust the SP in the dump header when copying the dump file
0000 64 so that it is right the next time through.
0000 65
0000 66 V03-001 KTA0093 Kerbey T. Altmann 05-Apr-1982
0000 67 Modifications to use PAGEFILE.SYS as dumpfile.
0000 68
0000 69 V02-007 KDM0063 Kathleen D. Morse 04-Aug-1981
0000 70 Increment dump version number to 2.
0000 71
0000 72 V02-006 MTR0001 Mike Rhodes 22-Jun-1981
0000 73 Change default addressing mode to longword.
0000 74 Remove references to $$DAMSGDEF macro.
0000 75
0000 76 V02-005 KDM0041 Kathleen D. Morse 02-Mar-1981
0000 77 Remove local definitions for DMP$ symbols.
0000 78
0000 79 V02-004 TMH0004 Tim Halvorsen 01-Mar-1981
0000 80 Fix ASSUME in processing memory controller descriptors.
0000 81
0000 82 V02-003 TMH0003 Tim Halvorsen 10-Feb-1981
0000 83 Change severity on REQMEM status from severe to error.
0000 84 to avoid having image exit.
0000 85 Do not report "file locked by another user" errors when

```

```
0000 86 : marking dump file analyzed.
0000 87 :
0000 88 : V02-002 TMH0002 Tim Halvorsen 19-Jan-1981
0000 89 : Allow dumps which are not long enough to contain all
0000 90 : memory on the system as long as it contains the system
0000 91 : page table. Issue warning message when dump file isn't
0000 92 : quite long enough, giving the number of blocks it should be.
0000 93 :
0000 94 : V02-001 TMH0001 Tim Halvorsen 19-Oct-1980
0000 95 : Support dumps from systems with 2 discontinuous memory
0000 96 : controllers.
0000 97 :--
```

0000	99	.SBTTL	DECLARATIONS	
0000	100	:		
0000	101	:		
0000	102	:		
0000	103	\$STSDEF		: STATUS FIELD DEFINITIONS
0000	104	\$JPIDEF		: GETJPI DEFINITIONS
0000	105	\$SECDEF		: CRMPSC ARGUMENT DEFINITIONS
0000	106	\$DMPDEF		: DUMP FILE DEFINITIONS
0000	107	\$PRTDEF		: PROTECTION CODES
0000	108	\$PTEDEF		: PAGE TABLE ENTRY DEFINITIONS
0000	109	\$RPBDEF		: RESTART PARAMETER BLOCK
0000	110	\$VADEF		: VIRTUAL ADDRESS DEFINITIONS
0000	111	\$EMBDEF	CR	: ERROR MESSAGE BUFFER OFFSETS

```
0000 113 .SBTTL STORAGE DEFINITIONS
0000 114 :
0000 115 : STORAGE DEFINITIONS
0000 116 :
0000 117 :
00000000 118 .PSECT SDADATA,NOEXE,WRT
0000 119
00000004 0000 120 PHYS_PAGES::
0000 121 .BLKL 1 ; PHYSICAL MEMORY SIZE
0004 122
0004 123 AVL_RANGE:
00000200 0004 124 .LONG ^X200 ; STARTING ADDRESS (SPECIFY PO RANGE)
3FFFFFFF 0008 125 .LONG ^X3FFFFFFF ; ENDING ADDRESS
00000014 000C 126 MAPRANGE:
0014 127 .BLKL 2 ; STARTING,ENDING ADDRESS
0014 128
00000018 0014 129 MAPPED_SBR::
0018 130 .BLKL 1 ; ADDRESS OF SPT IN MAPPED AREA
0018 131
0000001C 0018 132 GETMEM_BUFFER::
001C 133 .BLKL 1 ; FOR 1 LONGWORD TRANSFERS
001C 134
00000020 001C 135 DEMAND_ZERO:
0020 136 .BLKL 1 ; ADDRESS OF DEMAND ZERO PAGE
0020 137
00000024 0020 138 POBR:: .BLKL 1 ; PO BASE REGISTER
00000028 0024 139 POLR:: .BLKL 1 ; PO LENGTH REGISTER
0000002C 0028 140 P1BR:: .BLKL 1 ; P1 BASE REGISTER
00000030 002C 141 P1LR:: .BLKL 1 ; P1 LENGTH REGISTER
0030 142
00000000 143 .PSECT MAPPING,EXE,NOWRT
0000 144
0000 145 .DEFAULT DISPLACEMENT, LONG
```

MAP
Sym
SS
SS
SS1
ALL
ARC
AVL
CUP
DEF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DMF
DUP
DUP
DUP
DUP
EME
EME
EXI
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FAE
FIL
GE
GE
GE
LI
LO
MAI
MAI
MAI
MAI
MAI
MAI
MSI
MSI
MSI
MSI


```

0000 147 .SBTTL MAP_DUMP - MAP THE DUMP INTO VIRTUAL MEMORY
0000 148 :---
0000 149 MAP_DUMP
0000 150
0000 151 THIS ROUTINE ATTEMPTS TO MAP THE DUMP FILE AS A PRIVATE
0000 152 SECTION INTO THE PROCESS REGION OF VIRTUAL MEMORY. IF
0000 153 THE MAPPING CANNOT BE DONE, AN ERROR IS RETURNED TO THE
0000 154 CALLER.
0000 155
0000 156 INPUTS:
0000 157
0000 158 NONE
0000 159
0000 160 OUTPUTS:
0000 161
0000 162 RO = SUCCESS/FAILURE FLAG
0000 163 IF SUCCESS, THE DUMP CAN NOW BE ACCESSED BY READING THE
0000 164 CORRESPONDING VIRTUAL MEMORY LOCATION.
0000 165 :---
0000 166
0000 167 .ENABL LSB
0000 168
023C 0000 169 .ENTRY MAP_DUMP, ^M<R2,R3,R4,R5,R9>
0002 170
04 00000000'EF E9 0002 171 BLBC CURRENT_SYSTEM,5$ ; BRANCH IF EXAMINING DUMP
50 01 D0 0009 172 MOVL #1,R0 ; SUCCESS
04 000C 173 RET ; IF CURRENT SYSTEM, EXIT
000D 174 5$:
52 00000000'EF DE 000D 175 MOVAL DUMPR,R2
0014 176 $READ (R2) ; READ DUMP HEADER (3 BLOCKS)
001D 177 SIGNAL RMS,(R2)
59 0000'C2 D0 0030 178 MOVL RAB$L_RBF(R2),R9 ; GET ADDRESS OF DUMP HEADER
52 00000000'EF DE 0035 179 MOVAL DUMPF,R2
003C 180 $CLOSE (R2) ; CLOSE DUMP FILE
0045 181 SIGNAL RMS,(R2)
6D 00000004'EF 01 E0 0058 182 BBS #DMP$V_EMPTY,DUMP_HEADER+DMP$L_FLAGS,15$
0060 183 ; LEAVE NOW IF DUMP IS EMPTY
0000'C2 00000000'8F D0 0060 184 MOVL #FAB$M_UFO,FAB$L_FOP(R2) ; USER FILE OPEN
0069 185 $OPEN (R2) ; RE-OPEN FILE FOR CRMPSC
0072 186 SIGNAL RMS,(R2)
02 06 A9 B1 0085 187 CMPW DMP$W_DUMPVER(R9),#2 ; VERSION MUST BE < 2
2F 14 0089 188 BGTR 10$ ; IF NOT, NOT A VALID DUMP FILE
50 68 A9 64 A9 CD 0088 189 XORL3 DMP$L_SYSVER(R9),DMP$L_CHECK(R9),R0 ; RO=(SYSVER XOR CHECK)
50 D6 0091 190 INCL R0 ; IS CHECK IS ONE'S COMP. OF SYSVER?
25 12 0093 191 BNEQ 10$ ; BRANCH IF NOT VALID
0095 192 ; THIS CODE ASSUMES THAT THE SYSTEM PAGE TABLE IS AT THE
0095 193 ; END OF MAIN PHYSICAL MEMORY.
53 D4 0095 194 CLRL R3 ; INIT PAGE COUNTER
0097 195 ASSUME DMP$C_NMEMDSC EQ RPB$C_NMEMDSC
54 08 9A 0097 196 MOVZBL #DMP$C_NMEMDSC,R4 ; MAX # OF MEMORY DESCRIPTORS
55 00000024'EF 9E 009A 197 MOVAB DUMP_HEADER+DMP$L_MEMDSC,R5 ; GET ADR OF MEMORY DESCRIPTORS
50 65 18 00 EF 00A1 198 7$: EXTZV #DMP$V_PAGCNT,#DMP$S_PAGCNT,(R5),R0 ; GET PAGE CNT FOR THIS MEM
09 13 00A6 199 BEQL 8$ ; BR IF NO MORE MEMORY DESCRIPTORS USED
53 50 C0 00A8 200 ADDL2 R0,R3 ; ACCUMULATE TOTAL # OF PAGES
00AB 201 ASSUME DMP$C_MEMDSCSIZ EQ RPB$C_MEMDSCSIZ
55 08 C0 00AB 202 ADDL2 #DMP$C_MEMDSCSIZ,R5 ; GET NEXT MEMORY DESCRIPTOR
F0 54 F5 00AE 203 SOBGTR R4,7$ ; LOOP ONCE FOR EACH MEMORY DESCRIPTOR

```

?SE

SAE

SD/

MAI

Phi

In

Con

Pa

Syn

Syn

Pse

Cre

Ass

The

532

The

674

38

Mac

TOI

830

The

MAI

```

00000200 8F 53 D1 00B1 204 8$: CMPL R3,#512 ; MUST BE AT LEAST 256K (1/4 MEG)
          1C 1E 0GB8 205 BGEQU 20$ ; BRANCH IF OK
          04 00BA 206 10$: SIGNAL 0,DUMPEMPTY ; SIGNAL NO VALID DUMP FOUND
          00CC 207 RET
          00CD 208
000C0000'EF 00 FB 00CD 209 15$: CALLS #0,EXIT_IF_OLD ; ONLY CALLING TO FLUSH INPUT
          E4 11 00D4 210 BRB 10$ ; LEAVE QUIETLY
          00D6 211 20$:
54 0000000C'EF DE 00D6 212 MOVAL MAPRANGE,R4
          00DD 213 $CRMPSC,S INADR=AVLRANGE, - ; MAP SECTION
          00DD 214 RETADR=(R4), - ; RESULT ADDRESS RANGE
          00DD 215 CHAN=FAB$L $TV(R2), - ; CHANNEL AS RETURNED BY OPEN
          00DD 216 FLAGS=#SECSM_EXPREG, - ; READABLE/EXPAND REGION SECTION
          00DD 217 PAGCNT=R3, - ; NUMBER OF PAGES TO MAP
          00DD 218 VBN=#4 ; STARTING BLOCK IN FILE
          0105 219
          52 04 A4 64 C3 0111 220 SIGNAL
          52 D6 0116 221 SUBL3 (R4),4(R4),R2 ; LENGTH MAPPED - 1
          52 52 F7 8F 78 0118 222 INCL R2 ; TOTAL LENGTH OF SECTION
00000000'EF 52 D0 011D 223 ASHL #-9,R2,R2 ; LENGTH OF SECTION IN PAGES
          53 52 D1 0124 224 MOVL R2,PHYS_PAGES ; SAVE LENGTH OF DUMP FILE
          16 18 0127 225 CMPL R2,R3 ; DO WE HAVE ENTIRE DUMP?
          53 DD 0129 226 BGEQ 30$ ; BRANCH IF OK
          52 DD 012B 227 PUSHL R3 ; LENGTH DESIRED
          012D 228 PUSHL R2 ; LENGTH SUCCESSFULLY MAPPED
          013F 229 SIGNAL 2,SHORTDUMP ; INSUFFICIENT DUMP FILE SPACE
          013F 230
          013F 231 : LOCATE THE SYSTEM PAGE TABLE IN THE DUMP FILE. IF THE
          013F 232 : DUMP FILE WASN'T LONG ENOUGH TO GET IT, ISSUE A FATAL ERROR.
          53 08 A9 F7 8F 78 013F 233 30$: ASHL #-9,DMP$L SBR(R9),R3 ; GET PFN OF SYSTEM PAGE TABLE
          04ED 30 0145 234 BSBW LOCATE_PFN ; LOCATE PFN WITHIN DUMP FILE
          10 50 E9 0148 235 BLBC R0,35$ ; BRANCH IF ERROR
          00000014'EF 57 D0 014B 236 MOVL R7,MAPPED_SBR ; SAVE ADDRESS OF MAPPED SPT
          00000000'EF 53 D1 0152 237 CMPL R3,PHYS_PAGES ; BLOCK WITHIN DUMP FILE?
          12 1B 0159 238 BLEQU 40$ ; BRANCH IF WITHIN RANGE
          015B 239 35$: SIGNAL 0,SPTNOTFND ; SYSTEM PAGE TABLE NOT DUMPED
          016D 240
          016D 241 : INITIALIZE A BLOCK OF ZEROS FOR DEMAND ZERO REQUESTS
          016D 242
          00000200 8F DD 016D 243 40$: PUSHL #512 ; LENGTH IN BYTES TO ALLOCATE
          00000000'EF 01 FB 0173 244 CALLS #1,ALLOCATE ; ALLOCATE STORAGE
          017A 245 SIGNAL ; SIGNAL IF ANY ERRORS
          0000001C'EF 51 D0 0186 246 MOVL R1,DEMAND_ZERO ; SAVE ADDRESS OF PAGE
          61 0200 8F 00 6E 00 2C 018D 247 MOVCS #0,(SP),#0,#512,(R1) ; USE AS DEMAND ZERO PAGE
          04 0195 248 RET
          0196 249
          0196 250 .DSABL LSB

```

```

0196 252 .SBTTL SAVE_DUMP, Save dump file into another file
0196 253
0196 254 :---
0196 255 :
0196 256 SAVE_DUMP - SAVE filespec Command
0196 257 :
0196 258 This command copies the entire contents of the dump
0196 259 file to another file specified by the first parameter
0196 260 of the command.
0196 261 :
0196 262 :---
0196 263
00007E00 0196 264 MAX_SIZE = 63*512 ; Max. size of I/O transfer
0196 265
007C 0196 266 .ENTRY SAVE_DUMP,-
0198 267 ^M<R2,R3,R4,R5,R6>
0198 268
1A 00000000'EF E9 0198 269 BLBC CURRENT_SYSTEM,5$ ; Branch if not running system
019F 270 SIGNAL 0,NOTCOPIED ; Signal syntax error - not allowed
01B1 271 STATUS SUCCESS ; exit to tparse w/success
04 01B8 272 RET
01B9 273
53 00000000'EF 9E 01B9 274 5$: MOVAB SAVDMP,R3 ; R3 = RAB for new file
52 0000'C3 00 01C0 275 MOVL RAB$FAB(R3),R2 ; R2 = FAB for new file
50 00000000'EF 9E 01C5 276 MOVAB FILE_DESC,R0 ; Address of filespec descriptor
0000'C2 60 90 01CC 277 MOVB (R0),FAB$B_FNS(R2) ; Set length of file spec.
0000'C2 04 A0 00 01D1 278 MOVL 4(R0),FAB$C_FNA(R2) ; Set address of file spec.
01D7 279 $CREATE (R2) ; Create new file
01E0 280 SIGNAL RMS,(R2)
01F3 281 $CONNECT (R3)
01FC 282 SIGNAL RMS,(R3)
0000'C3 00000000'EF 9E 020F 283 MOVAB DUMP_HEADER,RAB$RBF(R3) ; Set buffer address
0000'C3 0000'8F 00 0218 284 MOVW #DUMP_HEADER_LEN,RAB$W_RSZ(R3)
56 0000006C'EF 9E 021F 285 MOVAB DUMP_HEADER+DMP$CRASHERL,R6 ; SET ADDR OF ERROR LOG ENTRY
02 00000006'EF 03 19 022D 286 CMPW DUMP_HEADER+DMP$W_DUMPVER,#2 ; VMS V2 OR V3 FORMAT?
03 04 C0 022F 287 BLSS 6$ ; XFER IF V2 FORMAT
56 5C A6 9E 0232 288 ADDL2 #EMBSK_LENGTH,R6 ; ELSE POINT PAST HDR FOR V3 FORMAT
66 08 C2 0236 289 6$: MOVAB EMB$CR_SP(R6),R6 ; SET ADDRESS OF SAVED STACK POINTER
0239 290 SUBL2 #2*4,(R6) ; ADJUST THE STACK
66 08 C0 0242 291 $WRITE (R3) ; Write out dump header blocks
2A 50 E8 0245 292 ADDL2 #2*4,(R6) ; ADJUST BACK FOR ANYTHING FOLLOWING
50 DD 0248 293 BLBS R0,8$ ; IF LBS, WRITE WAS SUCCESSFUL
0000'C2 00000000'8F 00 024A 294 PUSHL R0 ; SAVE WRITE ERROR STATUS
50 8E 00 0253 295 MOVL #<FAB$M_DLT!FAB$M_NAM>,FAB$FOP(R2) ; DELETE FILE ON CLOSE
025C 296 $CLOSE (R2) ; CLOSE THE FILE
025F 297 MOVL (SP)+,R0 ; RESTORE WRITE ERROR STATUS
0000'C3 0000000C'EF 00 0272 298 SIGNAL RMS,(R3) ; REPORT WRITE ERROR STATUS
0000'C3 7E00 8F 00 027B 299 8$: MOVL MAPRANGE,RAB$RBF(R3) ; Set starting buffer address
56 00000000'EF 09 78 0282 300 MOVW #MAX_SIZE,RAB$W_RSZ(R3) ; Set to max. transfer size
028A 301 ASHL #9,PHYS_PAGES,R6 ; Get file size in bytes in R6
00007E00 8F 56 01 028A 302 10$: CMPL R6,#MAX_SIZE ; Less than full transfer left?
05 14 0291 303 BGTR 15$ ; Branch if not
0000'C3 56 00 0293 304 MOVW R6,RAB$W_RSZ(R3) ; Set size of last transfer
0298 305 $WRITE (R3) ; Write into output file
50 0000'C3 3C 02A1 306 15$: SIGNAL RMS,(R3)
02B4 307 MOVZWL RAB$W_RSZ(R3),R0 ; Get length just transferred

```

SAVE_DUMP, Save dump file into another f 5-SEP-1984 03:33:07 [SDA.SRC]MAPPING.MAR;1

0000	'C3	50	C0	02B9	309	ADDL	R0,RAB\$RBF(R3)	:	Increment buffer address
	56	50	C2	02BE	310	SUBL	R0,R6	:	Subtract from loop count
		C7	14	02C1	311	BGTR	10\$:	Continue until done
				02C3	312	%CLOSE	(R2)	:	Close output file
				02CC	313	SIGNAL	RMS,(R2)	:	
50	00000000	'GF	DE	02DF	314	.WEAK	SDA\$RELEASE_DUMP	:	Do not force this in
		16	13	02DF	315	MOVAL	G^SDA\$RELEASE_DUMP,R0	:	See if it's there
	00000000	'EF	DD	02E6	316	BZL	20\$:	No, leave
	60	01	FB	02E8	317	PUSHL	DUMPF+FAB\$R_NAM	:	Yes, pass address of NAM block
	50	00'	D1	02EE	318	CALLS	#1,(R0)	:	to the routine
		08	12	02F1	319	CMPL	S^#SS\$_WASSET,R0	:	Did it return the blocks?
00	00000004	'EF	E2	02F4	320	BNEQ	20\$:	No, leave
		01	04	02F6	321	BBSS	#DMP\$V_EMPTY,DUMP_HEADER+DMP\$R_FLAGS,20\$:	Yes, set the bit
			04	02FE	322	RET	20\$:		

```

02FF 324 .SBTTL MARK_DUMP -- MARK DUMP ANALYZED
02FF 325
02FF 326
02FF 327 MARK_DUMP
02FF 328
02FF 329 SET A FLAG IN THE DUMP FILE TO INDICATE THAT THE
02FF 330 DUMP HAS BEEN ANALYZED AT LEAST ONCE.
02FF 331
02FF 332 INPUTS:
02FF 333
02FF 334 DUMP IS STILL MAPPED.
02FF 335
02FF 336 OUTPUTS:
02FF 337
02FF 338 DUMP IS UNMAPPED AND FILE IS CLOSED.
02FF 339
02FF 340
02FF 341
001C 02FF 342 .ENTRY MARK_DUMP,^M<R2,R3,R4>
0301 343
54 00000000'EF DE 0301 344 MOVAL DUMP_HEADER,R4
06 04 A4 01 E0 0308 345 BBS #DMP$V_EMPTY,DMP$L_FLAGS(R4),10$ ; Get rid of it if empty
01 04 A4 00 E1 030D 346 BBC #DMP$V_OLDDUMP,DMP$L_FLAGS(R4),10$
04 0312 347 RET
0313 348 10$:
0313 349 $DELTVAS MAPRANGE ; UNMAP SECTION
0324 350 SIGNAL
52 00000000'EF DE 0330 351 MOVAL DUMPF,R2
53 00000000'EF DE 0337 352 MOVAL DUMPR,R3
033E 353 $DASSGN_S FAB$L_STV(R2) ; DEASSIGN CHANNEL
034A 354 SIGNAL
0000'C2 0000'C2 D4 0356 355 CLRL FAB$L_FOP(R2) ; CLEAR UFO OPTION
0000'C2 00'8F 90 035A 356 MOVB #FAB$M_BIO!FAB$M_GET!FAB$M_PUT,FAB$B_FAC(R2)
0000'8F 50 B1 0360 357 $OPEN (R2) ; RE-OPEN DUMP FILE
0000'8F 07 13 0369 358 CMPW R0,#RMS$_PRV&^XFFFF ; PRIVILEGE VIOLATION?
0000'8F 50 B1 036E 359 BEQL 15$ ; SKIP IF NO PRIVILEGE
01 12 0370 360 CMPW R0,#RMS$_FLK&^XFFFF ; FILE LOCKED BY ANOTHER USER?
04 0375 361 BNEQ 20$ ; SKIP UPDATE IF SO
0377 362 15$:
0378 363 RET
038B 364 20$:
0394 365 SIGNAL RMS,(R2)
03A7 366 $CONNECT (R3)
03AC 367 SIGNAL RMS,(R3)
03B1 368 MOVL #1,RAB$L_BKT(R3) ; READ BLOCKS 1-3
03BA 369 MOVL R4,RAB$L_UBF(R3) ; SET BUFFER ADDRESS
03BF 370 MOVL #DUMP_HEADER_LEN,RAB$W_USZ(R3) ; AND LENGTH
03C8 371 BISL3 #<1@DMP$V_OLDDUMP>,- ; NOTE DUMP ANALYZED
03DB 372 $READ (R3) ; AND SAVE POSSIBLE EMPTY FLAG
03DF 373 SIGNAL RMS,(R3) ; RE-READ DUMP HEADER
03E8 374 POPL DMP$L_FLAGS(R4) ; RESTORE OLD COPY OF FLAGS
03FB 375 $WRITE (R3) ; RE-WRITE HEADER
0404 376 SIGNAL RMS,(R3)
0417 377 $CLOSE (R2) ; CLOSE FILE FOR GOOD
0417 378 RET

```

```

0418 380 .SBTTL GETMEM - READ DUMP MEMORY AREA
0418 381 :---
0418 382 :
0418 383 : GETMEM
0418 384 :
0418 385 : THIS ROUTINE TRANSFERS AN AREA FROM THE MEMORY IN THE
0418 386 : DUMP FILE TO THE CALLERS RETURN BUFFER. IT PERFORMS
0418 387 : THE NECESSARY ADDRESS TRANSLATION TO LOCATE THE DATA
0418 388 : IN THE DUMP FILE.
0418 389 :
0418 390 : INPUTS:
0418 391 : 0(AP) = NUMBER OF LONGWORD ARGUMENTS
0418 392 : 4(AP) = STARTING VIRTUAL ADDRESS IN DUMP
0418 393 : 8(AP) = (OPTIONAL) RETURN BUFFER ADDRESS
0418 394 : 12(AP) = (OPTIONAL) LENGTH OF TRANSFER, DEFAULT=4
0418 395 :
0418 396 : POBR-P1LR MUST BE SET IF ANY P0 OR P1 ADDRESSES
0418 397 : ARE TO BE TRANSLATED.
0418 398 :
0418 399 : OUTPUTS:
0418 400 :
0418 401 : R0 = SUCCESS IF BUFFER FOUND AND TRANSFERRED,
0418 402 : FAILURE IF ADDRESS NOT VALID OR NOT AVAILABLE.
0418 403 : R1 = FIRST LONGWORD OF MEMORY RETRIEVED.
0418 404 :
0418 405 :---
0418 406 :
0000 0418 407 .ENTRY GETMEM,0
7D'AF 6C FA 041A 408 CALLG (AP),B^TRYMEM ; ATTEMPT TO READ MEMORY
1E 50 E8 041E 409 BLBS R0,90$ ; BRANCH IF SUCCESSFUL
00000000'8F 50 D1 0421 410 CMPL R0,#SS$_NOPRIV ; NOT ENOUGH PRIVILEGE?
46 13 0428 411 BEQL OTHER ; BRANCH IF SO
04 AC DD 042A 412 PUSHL 4(AP) ; ADDRESS UNABLE TO READ
042D 413 SIGNAL 1,NOREAD ; WRITE WARNING MESSAGE
04 043F 414 90$: RET
0440 415
0000 0440 416 .ENTRY REQMEM,0
7D'AF 6C FA 0442 417 CALLG (AP),B^TRYMEM ; ATTEMPT TO READ MEMORY
26 50 E8 0446 418 BLBS R0,90$ ; BRANCH IF SUCCESSFUL
00000000'8F 50 D1 0449 419 CMPL R0,#SS$_NOPRIV ; NOT ENOUGH PRIVILEGE?
1E 13 0450 420 BEQL OTHER ; BRANCH IF SO
04 AC DD 0452 421 PUSHL 4(AP) ; ADDRESS UNABLE TO READ
0455 422 STATUS NOREAD ; GET MESSAGE CODE
50 03 02 F0 045C 423 INSV #ST$K_ERROR,- ; CHANGE TO ERROR INSTEAD OF WARNING
045E 424 #ST$V_SEVERITY,#ST$S_SEVERITY,R0
0461 425 SIGNAL 1 ; WRITE WITH 1 ARGUMENT
04 046F 426 90$: RET
0470 427
0470 428 OTHER: SIGNAL ; SIGNAL OTHER MESSAGES
047C 429 RET
07FC 047D 430
047D 431 .ENTRY TRYMEM,-
047F 432 ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
047F 433
59 04 AC D0 047F 434 MOVL 4(AP),R9 ; GET STARTING LOCATION DESIRED
03 6C D1 0483 435 CMPL (AP),#3 ; CHECK ALL ARGUMENTS SPECIFIED
0C 18 0486 436 BGEQ 5$ ; BRANCH IF ALL THERE

```

53	00000018'EF	9E	0488	437	MOVAB	GETMEM_BJFFER,R3	:	USE TEMPORARY SCRATCH BUFFER
	58 04	DO	048F	438	MOVL	#4,R8	:	ONE LONGWORD
		11	0492	439	BRB	7\$:	
			0494	440			5\$:	
	53 08 AC	DO	0494	441	MOVL	8(AP),R3	:	GET DESTINATION ADDRESS
	58 0C AC	DO	0498	442	MOVL	12(AP),R8	:	GET LENGTH DESIRED
			049C	443			7\$:	
	5A 53	DO	049C	444	MOVL	R3,R10	:	SAVE START OF BUFFER
			049F	445			:	
			049F	446			:	
			049F	447			:	
03	59 02 1E	ED	049F	448	CMPZV	#30,#2,R9,#^B11	:	INTERNAL REG. ADDRESS SPACE?
		12	04A4	449	BNEQ	4\$:	BRANCH IF NOT
	59 59	3C	04A6	450	MOVZWL	R9,R9	:	GET OFFSET INTO PHD
59	00000000'EF	CO	04A9	451	ADDL	PHDADR,R9	:	BIAS BY PHD ADDRESS
			04B0	452			4\$:	
			04B0	453			:	
			04B0	454			:	
			04B0	455			:	
			04B0	456			:	
			04B0	457			:	
			04B0	458			:	
27	00000000'EF	E9	04B0	458	BLBC	CURRENT_SYSTEM,10\$:	EXAMINING CURRENT SYSTEM?
	00000000'EF	DD	04B7	459	PUSHL	PROC_PID	:	CURRENT PROCESS PID
		58	04BD	460	PUSHL	R8	:	LENGTH TO TRANSFER
		53	04BF	461	PUSHL	R3	:	DESTINATION ADDRESS
		59	04C1	462	PUSHL	R9	:	VIRTUAL ADDRESS
	00000000'EF	04	04C3	463	CALLS	#4,GETPROCMEM	:	GET PROCESS MEMORY
		2F 50	04CA	464	BLBS	R0,50\$:	BRANCH IF SUCCESSFUL
	00000000'8F	50	04CD	465	CMPL	R0,#SS\$_TIMEOUT	:	MEMORY REQUEST TIMED OUT?
		29	04D4	466	BNEQ	90\$:	BRANCH IF NOT
	00000000'EF	D4	04D6	467	CLRL	PROC_PID	:	RETURN TO CURRENT USER CONTEXT
			04DC	468			:	TO ALLOW SYSTEM SPACE REQUESTS THRU
		21 11	04DC	469	BRB	90\$:	EXIT WITH STATUS
			04DE	470			:	
		58	04DE	471	PUSHL	R8	10\$:	LENGTH DESIRED
		59	04E0	472	PUSHL	R9	:	STARTING ADDRESS DESIRED
	61'AF	02	04E2	473	CALLS	#2,B^MAPMEM	:	PERFORM ADDRESS TRANSLATION
		16 50	04E6	474	BLBC	R0,90\$:	BRANCH IF ANY ERROR
63	67 56	28	04E9	475	MOVC	R6,(R7),(R3)	:	TRANSFER INTO USER BUFFER
	59 56	CO	04ED	476	ADDL2	R6,R9	:	INCREMENT VIRTUAL ADDRESS
	58 56	C2	04F0	477	SUBL2	R6,R8	:	DECREMENT LENGTH TO DO
		E9 14	04F3	478	BGTR	10\$:	LOOP UNTIL DONE
			04F5	479			:	
			04F5	480			:	
	51 6A	DO	04FC	481	STATUS	SUCCESS	50\$:	
		04	04FF	482	MOVL	(R10),R1	90\$:	RETURN FIRST WORD FOR FREE
					RET			

```

0500 484 .SBTTL PUTMEM, STORE INTO MAPPED MEMORY RANGE
0500 485 :---
0500 486 :
0500 487 : THIS IS USED TO STORE INTO A GIVEN DUMP MEMORY RANGE
0500 488 : SO THAT A SVPCTX CAN BE SIMULATED FROM THE CRASH
0500 489 : REGISTERS INTO THE PROCESS'S HARDWARE PCB.
0500 490 :
0500 491 : INPUTS:
0500 492 :
0500 493 : 4(AP) = ADDRESS IN DUMP MEMORY
0500 494 : 8(AP) = ADDRESS IN LOCAL MEMORY
0500 495 : 12(AP) = LENGTH OF TRANSFER
0500 496 :
0500 497 : OUTPUTS:
0500 498 :
0500 499 : R0 = STATUS CODE
0500 500 :
0500 501 :---
0500 502 :
07FC 0500 503 .ENTRY PUTMEM,-
0502 504 ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
0502 505
5A 04 AC DO 0502 506 MOVL 4(AP),R10 ; DESTINATION ADDRESS
59 08 AC DO 0506 507 MOVL 8(AP),R9 ; SOURCE ADDRESS
58 0C AC DO 050A 508 MOVL 12(AP),R8 ; LENGTH TO DO
050E 509 :
050E 510 : MAP INTERNAL REGISTER ADDRESS SPACE
050E 511 :
03 5A 02 1E ED 050E 512 CMPZV #30,#2,R10,#^B11 ; INTERNAL REGISTER SPACE?
0513 513 BNEQ 5$ ; BRANCH IF NOT
5A 5A 5A 3C 0515 514 MOVZWL R10,R10 ; GET OFFSET INTO PHD
5A 00000000'EF CO 0518 515 ADDL PHDADR,R10 ; MAP INTO PROCESS PHD
051F 516 5$:
051F 517 :
051F 518 : TRANSFER INTO DUMP MEMORY
051F 519 :
61'AF 58 DD 051F 520 10$: PUSHL R8 ; LENGTH DESIRED
5A DD 0521 521 PUSHL R10 ; DUMP ADDRESS
02 FB 0523 522 CALLS #2,B^MAPMEM ; MAP THE ADDRESS RANGE
7E 57 36 50 E9 0527 523 BLBC R0,90$ ; BRANCH IF ERROR
56 C1 052A 524 ADDL3 R6,R7,-(SP) ; SET ENDING ADDRESS
57 DD 052E 525 PUSHL R7 ; SET BEGINNING ADDRESS
55 5E DO 0530 526 MOVL SP,R5 ; MARK THE LOCATION
0533 527 $SETPRT_S INADR=(R5),- ; DESCRIPTOR
0533 528 PROT=#PRT$C_UW ; USER WRITABLE
5E 08 CO 0544 529 ADDL #8,SP ; CLEAN ADDRESS RANGE OFF STACK
16 50 E9 0547 530 BLBC R0,90$ ; LEAVE IF ERROR SO NO ACCVIO
67 69 56 28 054A 531 MOVC R6,(R9),(R7) ; TRANSFER INTO DUMP MEMORY
5A 56 CO 054E 532 ADDL R6,R10 ; INCREMENT DESTINATION ADDRESS
59 56 CO 0551 533 ADDL R6,R9 ; INCREMENT SOURCE ADDRESS
58 56 C2 0554 534 SUBL R6,R8 ; DECREMENT LENGTH
0557 535 BGTR 10$ ; BRANCH IF MORE TO DO
0559 536 STATUS SUCCESS
0560 537 90$:
0560 538 RET

```



```

0561 540 .SBTTL MAPMEM, MAP A GIVEN ADDRESS RANGE INTO LOCAL MEMORY
0561 541 :---
0561 542 :
0561 543 THIS ROUTINE PERFORMS ALL NECESSARY ADDRESS TRANSLATION
0561 544 IN ORDER TO REFERENCE A GIVEN RANGE OF DUMP MEMORY.
0561 545 :
0561 546 INPUTS:
0561 547 :
0561 548 4(AP) = STARTING ADDRESS OF DUMP MEMORY
0561 549 8(AP) = LENGTH OF DESIRED RANGE
0561 550 :
0561 551 OUTPUTS:
0561 552 :
0561 553 R0 = STATUS CODE
0561 554 R7 = ADDRESS IN LOCAL VIRTUAL MEMORY OF DUMP MEMORY
0561 555 R6 = LENGTH THAT CAN BE SUCCESSFULLY REFERENCED
0561 556 IN LOCAL MEMORY BEFORE ANOTHER TRANSLATION
0561 557 MUST BE DONE (END OF PAGE BOUNDARY).
0561 558 :---
0561 559 .ENABL LSB
0561 560
003C 0561 561 .ENTRY MAPMEM, ^M<R2,R3,R4,R5>
0563 562
52 54 04 AC D0 0563 563 MOVL 4(AP),R4 ; GET STARTING ADDRESS
56 08 AC D0 0567 564 MOVL 8(AP),R6 ; PRESET LENGTH TO TRANSFER
54 15 09 EF 056B 565 EXTZV #VASV_VPN,#VASS_VPN,R4,R2 ; VIRTUAL PAGE NUMBER
53 56 54 C1 0570 566 ADDL3 R4,R6,R3 ; ENDING ADDRESS + 1
53 53 15 09 EF 0574 567 DECL R3 ; COMPUTE ENDING ADDRESS
53 53 53 09 EF 0576 568 EXTZV #VASV_VPN,#VASS_VPN,R3,R3 ; GET VPN OF ENDING ADDRESS
53 00000200 8F 54 C1 057B 569 CMPL R2,R3 ; IS IT IN THE SAME PAGE?
53 000001FF 8F CA 057E 570 BEQL 20$ ; BRANCH IF SO
53 56 53 54 C3 0580 571 ADDL3 R4,#<1@VASV_VPN>,R3 ; INCREMENT VPN OF ADDRESS
53 0000020'EF 52 D1 0588 572 BICL2 #^X1FF,R3 ; COMPUTE ADDRESS OF NEXT PAGE
53 0000020'FF42 61 18 058F 573 SUBL3 R4,R3,R6 ; RESET LENGTH TO REST OF PAGE
53 0000020'FF42 11 11 0593 574 20$:
41 54 1F E0 0593 575 BBS #VASV_SYSTEM,R4,50$ ; BRANCH IF SYSTEM REGION
13 54 1E E0 0597 576 BBS #VASV_P1,R4,30$ ; BRANCH IF P1 SPACE
0000024'EF 52 D1 0598 577 CMPL R2,P0[R] ; CHECK IF IN BOUNDS
53 0000020'FF42 11 11 05A2 578 BGEQ NOTVALID ; BRANCH IF NOT
53 0000020'FF42 11 11 05A4 579 MOVAL @POBR[R2],R3 ; ADDRESS OF POPTTE
53 0000020'FF42 11 11 05AC 580 BRB 40$
000002C'EF 52 D1 05AE 581 30$:
0000028'FF42 4E 19 05B5 582 CMPL R2,P1LR ; CHECK IF IN BOUNDS
0000028'FF42 11 11 05B7 583 BLSS NOTVALID ; BRANCH IF NOT LEGAL
0000028'FF42 11 11 05B7 584 MOVAL @P1BR[R2],R3 ; ADDRESS OF P1PTE
5E 04 C2 05BF 585 40$:
51 5E D0 05BF 586 SUBL #4,SP ; ALLOCATE RETURN BUFFER
05C2 587 MOVL SP,R1 ; (DO NOT WIPE OUT CALLER'S
05C5 588 ; GETMEM BUFFER! HAS PARTIAL
05C5 589 ; RESULTS IN IT
05C5 590 TRYMEM (R3),(R1),#<4> ; GET PTE
2F 50 E9 05D0 591 POPL R2 ; GET PTE LONGWORD IN R2
11 11 05D3 592 BLBC R0,NOTVALID ; IF NOT FOUND
05D6 593 BRB 60$
000000C'EF 52 D1 05D8 594 50$:
24 14 05D8 595 CMPL R2,DUMP_HEADER+DMP$SLR ; CHECK IF IN BOUNDS
24 14 05DF 596 BGTR NOTVALID ; IF NOT, THEN NOT VALID

```

```

52 0000014'FF42  D0 05E1 597      MOVL  @MAPPED_SBR[R2],R2      ; GET PAGE TABLE ENTRY
                    22 19 05E9 598 60$:  BLSS  70$                    ; BRANCH IF VALID
                    18 13 05E9 599      BEQL  NOTVALID                ; BRANCH IF NO ACCESS (NULL)
                    16 10 05EB 600      BBS   #PTESV_TYPO,R2,NOTVALID ; ALLOW TRANSITION/DZERO PAGES
                    15 0A 05F1 602      BBS   #PTESV_TYP1,R2,NOTVALID
53 52 15 00  EF 05F5 603      EXTZV #PTESV_PFN,#PTES_PFN,R2,R3 ; PFN=0 FOR DZERO PAGES
                    11 12 05FA 604      BNEQ  70$                    ; MAP PAGES IN TRANSITION
57 000001C'EF  D0 05FC 605      MOVL  DEMAND_ZERO,R7         ; SET ADDRESS OF ZERO PAGE
                    28 11 0603 606      BRB   80$
                    0605 607 NOTVALID:
                    0605 608          STATUS NOTVALID                ; RETURN ERROR
                    04 060C 609          RET
                    060D 610 70$:
53  F4 52 14  E0 060D 611      BBS   #PTES_PFN-1,R2,NOTVALID ; I/O PAGES ARE NOT VALID
    52 15 00  EF 0611 612      EXTZV #PTESV_PFN,#PTES_PFN,R2,R3 ; PHYSICAL PAGE NUMBER
                    1D 10 0616 613      BSBB  LOCATE_PFN              ; FIND PFN WITHIN DUMP FILE
                    EA 50  E9 0618 614      BLBC  R0,NOTVALID            ; ERROR IF PFN NOT FOUND IN DUMP
    00000000'EF 53  D1 061B 615      CMPL  R3,PHYS_PAGES          ; VALID BLOCK NUMBER?
                    E1 14 0622 616      BGTR  NOTVALID              ; WE GOT LOST
52  04 AC 09 00  EF 0624 617      EXTZV #VASV_BYTE,#VASS_BYTE,4(AP),R2 ; GET OFFSET INTO PAGE
                    57 52  C0 062A 618      ADDL  R2,R7                  ; RETURN MAPPED ADDRESS
                    062D 619 80$:
                    062D 620          STATUS SUCCESS                ; RETURN SUCCESSFUL
                    04 0634 621          RET
                    0635 622
                    0635 623          .DSABL LSB

```

```

0635 625 .SBTTL LOCATE_PFN, FIND PAGE WITHIN DUMP FILE
0635 626 :---
0635 627 :
0635 628 LOCATE A GIVEN PFN IN THE MAPPED DUMP FILE AND RETURN
0635 629 THE VIRTUAL BLOCK NUMBER (VBN) FROM THE START OF THE
0635 630 FIRST BLOCK DUMPED (NOT COUNTING THE DUMP HEADER BLOCKS).
0635 631 :
0635 632 : INPUTS:
0635 633 :
0635 634 R3 = PFN
0635 635 :
0635 636 : OUTPUTS:
0635 637 :
0635 638 R0 = TRUE IF MAPPED BY DESCRIPTORS, FALSE IF OUT OF RANGE
0635 639 R3 = VBN OF BLOCK CONTAINING SPECIFIED PAGE
0635 640 R7 = ADDRESS OF MAPPED PAGE IN VIRTUAL MEMORY
0635 641 :
0635 642 R0-R5 DESTROYED.
0635 643 :---
0635 644 :
0635 645 LOCATE_PFN:

```

```

                                52  D4 0635 646 CLRL R2 ; INITIALIZE ACCUMULATED PAGE COUNT
                                0637 647 ASSUME DMP$C NMEMDSC EQ RPB$C_NMEMDSC
55 54 08 9A 0637 648 MOVZBL #DMP$C_NMEMDSC,R4 ; # OF MEMORY CONTROLLER DESCRIPTORS
50 65 18 00 EF 063A 649 MOVAB DUMP HEADER+DMP$C MEMDSC,R5 ; GET ADR OF FIRST MEMORY DESCRIPTOR
                                1A 13 0646 650 72$: EXTZV #DMP$V_PAGCNT,#DMP$S_PAGCNT,(R5),R0 ; GET PAGE CNT FOR THIS MEM
                                57 04 A5 D0 0648 651 BEQL 76$ ; BR IF NO MORE MEMORY DESCRIPTORS USED
                                53 57 D1 064C 652 MOVL 4(R5),R7 ; GET BASE PFN FOR THIS MEMORY
                                57 08 14 064F 653 CMPL R7,R3 ; IS DESIRED PAGE IN THIS MEMORY?
                                57 50 C0 0651 654 BGTR 74$ ; BR ON NO, ADD IN PAGCNT & GET NXT MEM
                                57 53 D1 0654 655 ADDL2 R0,R7 ; GET PFN OF PAGE PAST THIS MEMORY
                                52 09 19 0657 656 CMPL R3,R7 ; IS DESIRED PAGE IN THIS MEMORY?
                                52 50 C0 0659 657 74$: BLSS 76$ ; BY ON YES, PAGE IS FOUND IN THIS MEM
                                55 08 C0 065C 658 74$: ADDL2 R0,R2 ; ACCUMULATE TOTAL # OF PAGES
                                DF 54 F5 065F 659 ASSUME DMP$C MEMDSCSIZ EQ RPB$C_MEMDSCSIZ
                                57 50 C2 0662 660 ADDL2 #DMP$C_MEMDSCSIZ,R5 ; NEXT MEMORY CONTROLLER DESCRIPTOR
                                53 57 C2 0665 661 76$: SOBGTR R4,72$ ; LOOP ONCE FOR EACH MEMORY DESCRIPTOR
                                53 13 19 0668 662 76$: SUBL2 R0,R7 ; GET BASE PFN FOR MEMORY
                                53 52 C0 066A 663 SUBL2 R7,R3 ; COMPUTE OFFSET TO PAGE W/IN MEMORY
57 52 52 53 09 78 066D 664 BLSS 80$ ; BRANCH IF NOT IN RANGE
50 50 01 D0 0671 665 ADDL2 R2,R3 ; CONVERT PFN TO VBN WITHIN MEMORY DUMP
                                05 067C 666 ASHL #9,R3,R2 ; CONVERT TO BYTE OFFSET
                                50 D4 067D 667 ADDL3 MAPRANGE,R2,R7 ; COMPUTE ADDRESS OF MAPPED PAGE
                                05 067F 668 MOVL #1,R0 ; SUCCESS
                                80$: CLRL R0 ; FAILURE - PFN NOT MAPPED BY DUMP

```

MAPPING
V04-000

DUMP MEMORY MAPPING ROUTINES H 13
LOCATE_PFN, FIND PAGE WITHIN DUMP FILE

16-SEP-1984 01:34:19 VAX/VMS Macro V04-00
5-SEP-1984 03:33:07 [SDA.SRC]MAPPING.MAR;1

Page 17
(13)

MM
V0

0680 673
0680 674 .END

MAPPING
Symbol table

DUMP MEMORY MAPPING ROUTINES

I 13

16-SEP-1984 01:34:19 VAX/VMS Macro V04-00
5-SEP-1984 03:33:07 [SDA.SRC]MAPPING.MAR;1

Page 18
(13)

MM
VO

\$\$TMP1	= 00000001		MSG\$ SHORTDUMP	*****	X	03
\$\$TMP2	= 00000062		MSG\$ SPTNOTFND	*****	X	03
\$\$T1	= 00000000		MSG\$ SUCCESS	*****	X	03
ALLOCATE	*****	X	NOTVALID	00000605	R	03
ARGS	= 00000003		OTHER	00000470	R	03
AVLRANGE	00000004	R	POBR	00000020	RG	02
CURRENT_SYSTEM	*****	X	POLR	00000024	RG	02
DEMAND_ZERO	0000001C	R	P1BR	00000028	RG	02
DMP\$C_MEMDSCSIZ	= 00000008		P1LR	0000002C	RG	02
DMP\$C_NMEMDSC	= 00000008		PHDADR	*****	X	03
DMP\$C_CHECK	= 00000068		PHYS_PAGES	00000000	RG	02
DMP\$C_CRASHERL	= 0000006C		PROC_PID	*****	X	03
DMP\$C_FLAGS	= 00000004		PRT\$C_UW	= 00000004		
DMP\$C_MEMDSC	= 00000024		PTE\$S_PFN	= 00000015		
DMP\$C_SBR	= 00000008		PTE\$V_PFN	= 00000000		
DMP\$C_SLR	= 0000000C		PTE\$V_TYPO	= 00000016		
DMP\$C_SYSVER	= 00000064		PTE\$V_TYP1	= 0000001A		
DMP\$S_PAGCNT	= 00000018		PUTMEM	00000500	RG	03
DMP\$V_EMPTY	= 00000001		RAB\$S_BKT	*****	X	03
DMP\$V_OLDDUMP	= 00000000		RAB\$S_FAB	*****	X	03
DMP\$V_PAGCNT	= 00000000		RAB\$S_RBF	*****	X	03
DMP\$W_DUMPVER	= 00000006		RAB\$S_UBF	*****	X	03
DUMPF	*****	X	RAB\$W_RSZ	*****	X	03
DUMPR	*****	X	RAB\$W_USZ	*****	X	03
DUMP_HEADER	*****	X	REQMEM	00000440	RG	03
DUMP_HEADER_LEN	*****	X	RM\$S_FLK	*****	X	03
EMBS\$C_LENCTR	= 00000004		RM\$S_PRV	*****	X	03
EMBS\$C_CR_SP	= 0000005C		RPB\$C_MEMDSCSIZ	= 00000008		
EXIT_IF_OLD	*****	X	RPB\$C_NMEMDSC	= 00000008		
FAB\$B_FAC	*****	X	SAVDMP	*****	X	03
FAB\$B_FNS	*****	X	SAVE_DUMP	00000196	RG	03
FAB\$B_FNA	*****	X	SDASRELEASE_DUMP	*****	W GX	03
FAB\$B_FOP	*****	X	SECSM_EXPREG	= 0020000		
FAB\$B_NAM	*****	X	SS\$ NOPRIV	*****	X	03
FAB\$B_STV	*****	X	SS\$ TIMEOUT	*****	X	03
FAB\$M_BIO	*****	X	SS\$ WASSET	*****	X	03
FAB\$M_DLT	*****	X	ST\$K_ERROR	= 00000002		
FAB\$M_GET	*****	X	ST\$S\$ SEVERITY	= 00000003		
FAB\$M_NAM	*****	X	ST\$S\$V SEVERITY	= 00000000		
FAB\$M_PUT	*****	X	SYSS\$CLOSE	*****	GX	03
FAB\$M_UFO	*****	X	SYSS\$CONNECT	*****	GX	03
FILE_DESC	*****	X	SYSS\$CREATE	*****	GX	03
GETMEM	00000418	RG	SYSS\$CRMPSC	*****	GX	03
GETMEM_BUFFER	00000018	RG	SYSS\$DASSGN	*****	GX	03
GETPRO\$MEM	*****	X	SYSS\$DELTVA	*****	GX	03
LIB\$SIGNAL	*****	X	SYSS\$OPEN	*****	GX	03
LOCATE_PFN	00000635	R	SYSS\$READ	*****	GX	03
MAPMEM	00000561	RG	SYSS\$SETPRT	*****	GX	03
MAPPED_SBR	00000014	RG	SYSS\$WRITE	*****	GX	03
MAPRANGE	0000000C	R	TRYMEM	0000047D	RG	03
MAP_DUMP	00000000	RG	VAS\$ BYTE	= 00000009		
MAR_DUMP	000002FF	RG	VAS\$ VPN	= 00000015		
MAX_SIZE	= 00007E00		VAS\$V_BYTE	= 00000000		
MSG\$ DUMPEMPTY	*****	X	VAS\$V_P1	= 0000001E		
MSG\$ NOREAD	*****	X	VAS\$V_SYSTEM	= 0000001F		
MSG\$ NOTCOPIED	*****	X	VAS\$V_VPN	= 00000009		
MSG\$ NOTVALID	*****	X				

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SDADATA	00000C30 (48.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
MAPPING	00000680 (1664.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.05	00:00:02.93
Command processing	107	00:00:00.48	00:00:05.16
Pass 1	293	00:00:06.41	00:00:27.44
Symbol table sort	0	00:00:00.58	00:00:01.94
Pass 2	134	00:00:01.59	00:00:07.40
Symbol table output	14	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	581	00:00:09.19	00:00:45.95

The working set limit was 1650 pages.
53265 bytes (105 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 636 non-local and 64 local symbols.
674 source lines were read in Pass 1, producing 43 object records in Pass 2.
38 pages of virtual memory were used to define 36 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SDA.OBJ]SDALIB.MLB;1	3
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	7
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	23
TOTALS (all libraries)	33

836 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:MAPPING/OBJ=OBJ\$:MAPPING MSRC\$:MAPPING/UPDATE=(ENH\$:MAPPING)+EXECMLS/! IB+LIB\$:SDALIB/LIB

A large grid of 10 columns and 10 rows of small, faint technical diagrams and code snippets. The grid is densely packed with these elements, which appear to be fragments of system documentation or code listings. Several larger, more legible labels are scattered throughout the grid, including:

- HANDLER LIS
- DUMP LIS
- EXAMPS LIS
- INDEX LIS
- LOCK LIS
- MAIN LIS
- MAPPING LIS
- MMG LIS
- PARSE LIS

The overall appearance is that of a high-resolution scan of a printed document, where the individual grid cells contain small-scale technical information.