

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	

```

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          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

LOADER
Table of contents

N 13
- LOAD A DRIVER AND/OR DEVICE CONTROL BL 15-SEP-1984 23:54:17 VAX/VMS Macro V04-00

Page 0

(1) 102
(1) 164
(1) 260
(1) 291
(1) 651

DECLARATIONS
LOADER - LOAD A DEVICE DRIVER/DATABASE
CHECK DRV - CHECK IF DRIVER ALREADY LOADED
LOAD DB - LOAD THE DATABASE
DB_ERROR - ERROR LOADING DATABASE

```

0000 1      .TITLE  LOADER - LOAD A DRIVER AND/OR DEVICE CONTROL BLOCKS
0000 2      .IDENT  'V04-000'
0000 3
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 28
0000 29     :++
0000 30     : FACILITY:      SYSGEN
0000 31
0000 32     : ABSTRACT:      LOAD A DRIVER AND/OR DEVICE CONTROL BLOCKS
0000 33
0000 34
0000 35     : ENVIRONMENT:  USER MODE PRIVILEGED CODE
0000 36
0000 37     : AUTHOR:        LEN KAWELL, CREATION DATE:16-JUN-1978
0000 38
0000 39     : MODIFICATION HISTORY:
0000 40
0000 41     : V03-015 WHM0005      Bill Matthews      19-Jul-1984
0000 42     : Don't issue vector in use error message if the vector has
0000 43     : been in use by the console terminal driver(CON$INTINP).
0000 44
0000 45     : V03-014 LMP0275      L. Mark Pilant,      12-Jul-1984 12:10
0000 46     : Initialize the ACL info in the ORB to be a null descriptor
0000 47     : list rather than an empty queue. This avoids the overhead
0000 48     : of locking and unlocking the ACL mutex, only to find out
0000 49     : that the ACL was empty.
0000 50
0000 51     : V03-013 LMP0221      L. Mark Pilant,      31-Mar-1984 10:46
0000 52     : Add support for an ORB, the Object's Rights Block.
0000 53
0000 54     : V03-012 WHM0004      Bill Matthews      16-Feb-1984
0000 55     : Added part 2 of a 2 part change to clean up the support of
0000 56     : combo style devices.
0000 57

```

0000	58	:	V03-011 WHM0003	Bill Matthews	04-Feb-1984
0000	59	:	Added part 1 of a 2 part change to clean up the support		
0000	60	:	of combo style devices.		
0000	61	:			
0000	62	:	V03-010 LMP0185	L. Mark Pilant,	26-Jan-1984 15:28
0000	63	:	Add support for ACLs on devices.		
0000	64	:			
0000	65	:	V03-009 WHM0002	Bill Matthews	03-Jan-1984
0000	66	:	In LOAD DB don't assume the I/O database hasn't been		
0000	67	:	modified since SYSGEN called SGN\$GET_DEVICE. The CONFIGURE		
0000	68	:	process could have altered the I/O database. ACF\$GL_LASTDDB		
0000	69	:	isn't necessarily the last DDB and the UCB may have been		
0000	70	:	created.		
0000	71	:			
0000	72	:	V03-008 WHM0001	Bill Matthews	14-Dec-1983
0000	73	:	Use IDB\$B_COMBO_VECTOR to replace IDB\$B_VECTOR functionality.		
0000	74	:			
0000	75	:	V03-007 ROW0221	Ralph O. Weber	8-SEP-1983
0000	76	:	In LOAD DB guarantee that a success status is available to		
0000	77	:	allow IOC\$UNITINIT to be called, even if the call to		
0000	78	:	IOGEN\$CNTRL_INI is skipped.		
0000	79	:			
0000	80	:	V03-006 ROW0203	Ralph O. Weber	5-AUG-1983
0000	81	:	Change LOAD_DB to use IOC\$CTRLINIT, the common, system-wide		
0000	82	:	routine for calling driver's unit initialization routines.		
0000	83	:			
0000	84	:	V03-005 TCM0001	Trudy C. Matthews	01-Jun-1983
0000	85	:	Initialize the DDB\$L_ALLOCLS (device allocation class field)		
0000	86	:	in routine CREATE_DDB.		
0000	87	:			
0000	88	:	V03-004 MSH0004	Maryann Hinden	09-Feb-1983
0000	89	:	Add support for cluster device names.		
0000	90	:			
0000	91	:	V03-003 MSH0003	Maryann Hinden	06-Oct-1982
0000	92	:	Modify MSH0002.		
0000	93	:			
0000	94	:	V03-002 MSH0002	Maryann Hinden	05-Oct-1982
0000	95	:	Change check for zero UCB pointer when loading		
0000	96	:	database.		
0000	97	:			
0000	98	:	V03-001 MSH0001	Maryann Hinden	31-JUL-1982
0000	99	:	Add check for missing DDT's.		
0000	100	--			

```

0000 102      .SBTTL  DECLARATIONS
0000 103
0000 104
0000 105 :
0000 106 : INCLUDE FILES:
0000 107 :
0000 108 :
0000 109 :
0000 110 : MACROS:
0000 111 :
0000 112 :
0000 113 :
0000 114 : EQUATED SYMBOLS:
0000 115 :
0000 116 :
0000 117      $SYSGMSGDEF      ;DEFINE SYSGEN MESSAGES
0000 118      $IPLDEF          ;DEFINE SYSTEM IPL'S
0000 119      $DCDEF          ;DEFINE ADAPTER TYPE SYMBOLS
0000 120      $DYNDEF          ;DEFINE DYNAMIC MEMORY TYPES
0000 121      $DDBDEF          ;DEFINE DEVICE DATA BLOCK
0000 122      $UCBDEF          ;DEFINE UNIT CONTROL BLOCK
0000 123      $CRBDEF          ;DEFINE CHANNEL CONTROL BLOCK
0000 124      $VECDDEF          ;DEVICE INTERRUPT VECTOR
0000 125      $IDBDEF          ;DEFINE INTERRUPT DISPATCH BLOCK
0000 126      $ADPDEF          ;DEFINE ADAPTER CONTROL BLOCK
0000 127      $PRDEF          ;DEFINE PROCESSOR REGISTERS
0000 128      $DPTDEF          ;DEFINE DRIVER PROLOGUE TABLE
0000 129      $ACFDEF          ;DEFINE AUTO-CONFIGURE ARG LIST
0000 130      $PTEDEF          ;DEFINE PAGE TABLE ENTRIES
0000 131      $UBADEF          ;DEFINE UNIBUS ADAPTER
0000 132      $DDTDEF          ;DEFINE DDT
0000 133      $ORBDEF          ;DEFINE OBJECT'S RIGHTS BLOCK OFFSETS
0000 134 :
0000 135 : OWN STORAGE:
0000 136 :
0000 137 :
00000000 138      .PSECT  NONPAGED_DATA      rd,wrt,noexe,quad
0000 139
00000001 140  LOAD_FLAGS:      ;CONTROL BLOCKS CREATED FLAGS
0000 141      .BLKB      1
0001 142      _VIELD      LOAD,0,<-
0001 143      <DDB,,M>,-      ; FLAG DEFINITIONS
0001 144      <CRB,,M>,-      ; DDB CREATED
0001 145      <IDB,,M>,-      ; CRB CREATED
0001 146      <UCB,,M>,-      ; IDB CREATED
0001 147      >              ; UCB CREATED
0001 148  DDB_BLINK:      ;DDB BACKWARD LINK
0001 149      .BLKL      1
0005 150  UCB_BLINK:      ;UCB BACKWARD LINK
0005 151      .BLKL      1
0009 152
00000000 153      .PSECT  NONPAGED_CODE      rd,nowrt,exe,long
0000 154
0000 155  MBINT_DISP:      ;MBA INTERRUPT DISPATCHER MODEL
0000 156      PUSHF      #^M<R2,R3,R4,R5>
0002 157      JSB      @#0
0008 158

```

LOADER
V04-000

- LOAD A DRIVER AND/OR DEVICE CONTROL BL 15-SEP-1984 23:54:17 VAX/VMS Macro V04-00
DECLARATIONS 4-SEP-1984 23:04:34 [BOOTS.SRC]LOADER.MAR;1

Page 4
(1)

00000000 3F BB 0008 159 INT_DISP: ;GENERAL INTERRUPT DISPATCHER MODEL
9F 16 0008 160 PUSHR #^M<R0,R1,R2,R3,R4,R5>
00CA 161 JSB @#0
0010 162

```

0010 164      .SBTTL  LOADER - LOAD A DEVICE DRIVER/DATABASE
0010 165
0010 166      :++
0010 167      : FUNCTIONAL DESCRIPTION:
0010 168      :
0010 169      :     This routine will load a device driver, a device driver and
0010 170      :     database, or just a single unit into an already existing
0010 171      :     database.
0010 172      :
0010 173      : CALLING SEQUENCE:
0010 174      :
0010 175      :     CALL IOGEN$LOADER(ACF_LIST)
0010 176      :
0010 177      : INPUT PARAMETERS:
0010 178      :
0010 179      :     ACF_LIST -
0010 180      :
0010 181      :     ACF$A_ADAPTER(AP) = ADDRESS OF ADAPTER CONTROL BLOCK
0010 182      :     ACF$A_CONFIGREG(AP) = ADDRESS OF CONFIGURATION STATUS REGISTER
0010 183      :     ACF$A_AVECTOR(AP) = OFFSET TO ADAPTER INTERRUPT VECTOR (SCB)
0010 184      :     ACF$B_AUNIT(AP) = ADAPTER UNIT NUMBER
0010 185      :     ACF$B_AFLAG(AP) = ADAPTER GENERATION CONTROL FLAGS
0010 186      :     ACF$A_CONTRLREG(AP) = ADDRESS OF CONTROL REGISTER
0010 187      :     ACF$A_CVECTOR(AP) = OFFSET TO CONTROLLER INTERRUPT VECTOR (TABLE)
0010 188      :     ACF$B_CUNIT(AP) = CONTROLLER UNIT NUMBER
0010 189      :     ACF$B_CNUMVEC(AP) = NUMBER OF CONTROLLER VECTORS
0010 190      :     ACF$A_DEVNAME(AP) = ADDRESS OF DEVICE NAME COUNTED STRING
0010 191      :     ACF$A_DRVNAME(AP) = ADDRESS OF DRIVER NAME COUNTED STRING
0010 192      :     ACF$B_COMBO_CSR_OFFSET(AP) = OFFSET BACK TO THE START OF THE CSRS FOR
0010 193      :     A COMBO DEVICE
0010 194      :     ACF$B_COMBO_VECTOR_OFFSET(AP) = OFFSET BACK TO THE START OF THE VECTORS FOR
0010 195      :     FOR A COMBO DEVICE
0010 196      :
0010 197      :
0010 198      : IMPLICIT INPUTS:
0010 199      :
0010 200      :     NONE
0010 201      :
0010 202      : OUTPUT PARAMETERS:
0010 203      :
0010 204      :     NONE
0010 205      :
0010 206      : IMPLICIT OUTPUTS:
0010 207      :
0010 208      :     DRIVER AND/OR DATABASE LOADED
0010 209      :
0010 210      : COMPLETION CODES:
0010 211      :
0010 212      :     RO = STATUS OF OPERATION
0010 213      :
0010 214      : SIDE EFFECTS:
0010 215      :
0010 216      :     NONE
0010 217      :
0010 218      : --
0010 219
0000 0010 220 .Entry  IOGEN$LOADER,0
  
```

```

0012 221 :
0012 222 : See if driver needs to be loaded
0012 223 :
17 0B AC 00 E0 0012 224 BBS #ACF$V_RELOAD,ACF$B_AFLAG(AP),10$ ;BR IF RELOAD REQUESTED
0000'8F 50 B1 0017 225 $CMKRNL_S W^CHECK_DRV,(AP) ;CHECK IF DRIVER LOADED
2C 13 0024 226 CMPW -R0,#SS$_NOPRIV ;NO PRIVILEGE?
0A 50 E8 0029 227 BEQLU 40$ ;BR IF YES
002E 228 BLBS R0,20$ ;BR IF DRIVER LOADED
002E 229
00000000'EF 18 AC DD 002E 230 10$: PUSHL ACF$L_DRVNAME(AP) ;SET ADDR OF DRIVER NAME
01 FB 0031 231 CALLS #1,IOGEN$LOADDRV ;LOAD THE DRIVER
0038 232
0038 233 ; Note that the errors from this call are suppressed if this is a load driver
0038 234 ; call only (if ACF$V_NOLOAD_DB is set). This is because the subsequent
0038 235 ; call from AUTOCONFIGURE takes this same code path and will report the
0038 236 ; error at that time. Unsupported devices do not print out the error message.
0038 237
17 03 E0 0038 238 20$: BBS #ACF$V_NOLOAD_DB,-
0B AC 003A 239 ACF$B_AFLAG(AP),30$ ;EXIT IF LOAD DRIVER CALL ONLY
18 50 E9 003D 240
003D 241 BLBC R0,50$ ;BR IF ERROR LOADING DRIVER
14 AC D5 0040 242
0F 13 0040 243 TSTL ACF$L_DEVNAME(AP) ;DEVICE NAME SPECIFIED?
0043 244 BEQL 30$ ;BR IF NOT - DON'T LOAD DATABASE
03 11 0045 245 $CMKRNL_S W^LOAD_DB,(AP) ;LOAD THE DATABASE
0052 246 BRB 40$ ;EXIT WITH STATUS
0054 247
50 01 D0 0054 248 30$: MOVL #1,R0
04 04 0057 249 40$: RET
0058 250
0058 251 :
0058 252 : Error occurred loading driver
0058 253 :
FA 04 E1 0058 254 50$: BBC #ACF$V_SUPPORT,-
0B AC 005A 255 ACF$B_AFLAG(AP),40$ ;BRANCH IF SUPPORTED DEVICE
F5 11 005D 256 BRB 30$ ;ALWAYS SUCCESS FOR UNSUPPORTED
005F 257
005F 258

```

```

005F 260 .SBTTL CHECK_DRV - CHECK IF DRIVER ALREADY LOADED
005F 261 :++
005F 262 :
005F 263 : Local kernel mode routine to check if the driver is already loaded.
005F 264 :
005F 265 :--
005F 266 CHECK_DRV:
005F 267 .WORD ^M<R2,R3,R4,R5,R6,R7>
55 18 AC 30 0061 268 BSBW IOGEN$LOCK_IODB ;LOCK THE I/O DATABASE
56 54 85 9A 0064 269 MOVL ACF$$_DRVNAME(AP),R5 ;GET ADDR OF DRIVER NAME
00000000'GF 9E 0068 270 MOVZBL (R5)+,R4 ;GET SIZE OF DRIVER NAME
57 56 D0 006B 271 MOVAB G^IOCSGL_DPTLIST,R6 ;GET ADDR OF DPT LIST
0072 272 MOVL R6,R7 ;SAVE IT
0075 273 10$:
56 50 D4 0075 274 CLRL R0 ;ASSUME NOT LOADED
57 66 D0 0077 275 MOVL DPT$_FLINK(R6),R6 ;GET ADDR OF NEXT DRIVER PROLOGUE
57 56 D1 007A 276 CMPL R6,R7 ;END OF LIST?
51 20 A6 9E 007D 277 BEQL 20$ ;BR IF YES
50 81 9A 007F 278 MOVAB DPT$_NAME(R6),R1 ;GET ADDR OF DRIVER NAME
65 54 00 61 50 2D 0083 279 MOVZBL (R1)+,R0 ;GET SIZE OF DRIVER NAME
0000'CF 56 D0 0086 280 CMPC5 R0,(R1),#0,R4,(R5) ;COMPARE DRIVER NAMES
50 01 D0 008C 281 BNEQ 10$ ;BR IF NOT EQUAL
008E 282 MOVL R6,W^ACF$GL_DPT ;SET ADDR OF DRIVER PROLOGUE
0093 283 MOVL #1,R0 ;SET SUCCESS
50 DD 0096 284 20$:
FF65' 30 0096 285 PUSHL R0 ;SAVE STATUS
50 8ED0 0098 286 BSBW IOGEN$UNLK_IODB ;UNLOCK THE I/O DATABASE
04 009B 287 POPL R0 ;RESTORE THE STATUS
009E 288 RET
009F 289

```

```

009F 291 .SBTTL LOAD_DB - LOAD THE DATABASE
009F 292 :++
009F 293 :
009F 294 : Local kernel mode routine to load the UCB and if not loaded yet,
009F 295 : the DDB, CRB, and IDB.
009F 296 :
009F 297 :--
009F 298 LOAD_DB:
009F 299 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5B 0000'CF 94 00A1 300 CLR B W^LOAD_FLAGS ;CLEAR LOADER FLAGS
0000'CF D0 00A5 301 MOVL W^ACF$GL_DPT,R11 ;GET ADDR OF DRIVER PROLOGUE
FF53' 30 00AA 302 BSBW IOGEN$LOCK_IDDB ;LOCK THE I/O DATABASE
00AD 303 :
00AD 304 : CHECK IF DEVICE DATABASE ALREADY LOADED
00AD 305 :
00AD 306 BBSC #ACF$V GETDONE,-
00AF 307 ACF$B_AFLAG(AP),2$ ; Skip the scan if done in CONNECT
7E 0D 0B AC E4 00B2 308 MOVZWL ACF$W_CUNIT(AP),-(SP) ; Push unit number
12 AC 3C 00B6 309 PUSHL ACF$L_DEVNAME(AP) ; Push device name address
14 AC DD 00B9 310 BSBW SGN$GET_DEVICE ; Get device addresses if they exist
FF44' 30 00BC 311 ADDL2 #8,SP ; Pop input off stack
5E 08 C0 00BF 312 :
00BF 313 2$: SETIPL #IPL$ SYNCH ; Synchronize I/O database access
5A 0000'CF D0 00C2 314 MOVL W^ACF$GL_DDB,R10 ; Address of DDB
2D 13 00C7 315 BEQL CREATE_DDB ; Branch if it doesn't exist
00C9 316 :
0000'CF D5 00C9 317 TSTL W^ACF$GL_UCB ; Is there a UCB for this unit?
03 13 00CD 318 BEQL 10$ ; Branch if no
0358 31 00CF 319 5$: BRW DB_EXIT ; Branch to common exit point
00D2 320 :
59 0000'CF D0 00D2 321 10$: MOVL W^ACF$GL_CRB,R9 ; Address of CRB
F6 13 00D7 322 BEQL 5$ ; If none, exit
58 0000'CF D0 00D9 323 MOVL W^ACF$GL_IDB,R8 ; Address of IDB
EF 13 00DE 324 BEQL 5$ ; If none, exit
57 04 AA D0 00E0 325 MOVL DDB$L_UCB(R10),R7 ; Get address of first UCB
0D 13 00E4 326 BEQL 30$ ; If eql no UCB'S go create one.
12 AC 54 A7 B1 00E6 327 20$: CMPW UCB$W_UNIT(R7),ACF$W_CUNIT(AP); Is UCB already loaded?
E2 13 00EB 328 BEQL 5$ ; If eql yes, nothing left to do
57 30 A7 D0 00ED 329 MOVL UCB$L_LINK(R7),R7 ; Get address of next UCB
F3 12 00F1 330 BNEQ 20$ ; If neq then there is another UCB
01D7 31 00F3 331 30$: BRW CREATE_UCB ; Create UCB for this unit
00F6 332 :
00F6 333 :
00F6 334 : CREATE THE DDB
00F6 335 :
00F6 336 CREATE_DDB:
5A 0000'CF D0 00F6 337 MOVL W^ACF$GL_LASTDDB,R10 ;Get address of last DDB in list
51 44 8F 9A 00FB 338 MOVZBL #DDB$K_LENGTH,R1 ;GET SIZE OF DDB
FEFE' 30 00FF 339 BSBW IOGEN$ALLOBLOCK ;CREATE THE DDB
03 50 E8 0102 340 BLBS R0,10$ ;BR IF SUCCESS
0329 31 0105 341 BRW DB_ERROR ;ELSE - ERROR
0000'CF 01 88 0108 342 10$: BISB #LOAD M DDB,W^LOAD_FLAGS ;SET DDB LOADED FLAG
08 A2 51 B0 010D 343 MOVL R1,DDB$W_SIZE(R2) ;SET SIZE
0A A2 06 90 0111 344 MOVB #DYN$C_DDB,DDB$B_TYPE(R2) ;SET TYPE
0001'CF 6A DE 0115 345 MOVAL DDB$L_LINK(R10),W^DDB_LINK ;SAVE DDB BACKWARD LINK
51 6A D0 011A 346 MOVL DDB$L_LINK(R10),R1 ;SAVE LINK TO NEXT DDB
6A 52 D0 011D 347 MOVL R2,DDB$L_LINK(R10) ;SET LINK TO THIS DDB

```

5A	52	D0	0120	348	MOVL	R2,R10	:GET ADDR OF DDB
6A	51	D0	0123	349	MOVL	R1,DDB\$LINK(R10)	:SET LINK TO NEXT DDB
	0B	96	0126	350	INCB	DPT\$B_REFC(R11)	:INCREMENT DPT REFERENCE COUNT
50	14	9A	0129	351	MOVZBL	@ACF\$C_DEVNAME(AP),R0	:GET SIZE OF DEVICE NAME
		D6	012D	352	INCL	R0	:ADD 1 BYTE FOR COUNT
14	AA	28	012F	353	MOVC	R0,@ACF\$C_DEVNAME(AP),DDB\$NAME(R10)	:SET DEVICE NAME
	50	9A	0135	354	MOVZBL	@ACF\$C_DRVNAME(AP),R0	:GET SIZE OF DRIVER NAME
		D6	0139	355	INCL	R0	:ADD 1 BYTE FOR COUNT
24	AA	28	013B	356	MOVC	R0,@ACF\$C_DRVNAME(AP),DDB\$DRVNAME(R10)	:SET DRIVER NAME
34	AA	D0	0141	357	MOVL	W*ACF\$G_SGL,DB\$SGL_SGL(R10)	:INSERT THE ADDRESS OF SYSTEM BLOCK
3C	AA	D0	0147	358	MOVL	G*CLUS\$G_ALLOCLS,-	:COPY THE DEVICE ALLOCATION CLASS
	00000000		014F	359		DDB\$ALLOCLS(R10)	:FROM THE SYSTEM-WIDE PARAMETER
			014F	360			
			014F	361			
			014F	362			
			014F	363			
59	6C	D0	014F	363	MOVL	ACF\$C_ADAPTER(AP),R9	:CONTAINS ADDR. OF CRB IF CRBBLT FLAG IS SE
	01	E0	0152	364	BBS	#ACF\$V_CRBBLT,-	
22	0B		0154	365		ACF\$B_AFLAG(AP),20\$	:BR. IF CRB & IDB ARE BUILT
	00	E0	0157	366	BBS	#DPT\$V_SUBCNTRL,-	
24	0D		0159	367		DPT\$B_FLAGS(R11),CREATE_CRB	:BR IF SUBCONTROLLER
	AB		015C	368			
0C	AB	91	015C	369	CMPB	#AT\$MBA,DPT\$B_ADPTYPE(R11)	:MBA ADAPTER?
	0E	13	0160	370	BEQL	15\$	:BRANCH IF YES
0C	AB	91	0162	371	CMPB	#AT\$DR,DPT\$B_ADPTYPE(R11)	:DR ADAPTER?
	02	13	0166	372	BEQL	15\$	:BRANCH IF YES
0C	AB	91	0168	373	CMPB	#AT\$CI,DPT\$B_ADPTYPE(R11)	:CI ADAPTER?
	04	13	016C	374	BEQL	15\$	:BRANCH IF YES
	02	11	016E	375	BRB	CREATE_CRB	:BRANCH IF OTHER
	10		0170	376			
			0170	377			
			0170	378			
			0170	379	15\$: MOVL	ACF\$C_ADAPTER(AP),R0	:GET ADDR OF ADP
59	50	D0	0173	380	MOVL	ADP\$C_CRB(R0),R9	:GET ADDR OF EXISTING CRB
	10	13	0177	381	BEQL	CREATE_CRB	:BRANCH IF NONE
58	2C	D0	0179	382	20\$: MOVL	CRB\$C_INTD+VEC\$C_IDB(R9),R8	:GET ADDR. OF IDB
	014D	31	017D	383	BRW	CREATE_UCB	:CREATE A NEW UCB
			0180	384			
			0180	385			
			0180	386			
			0180	387			
			0180	388	CREATE_CRB:		
51	1E	9A	0180	388	MOVZBL	ACF\$B_CNUMVEC(AP),R1	:GET NUMBER OF INT VECTORS
	51	D7	0184	389	DECL	R1	:ONE IS ALWAYS ASSUMED
	24	A4	0186	390	MULW	#VEC\$C_LENGTH,R1	:COMPUTE SIZE OF EXTRA DISPATCHERS
51	0048	A0	0189	391	ADDW	#CRB\$C_LENGTH,R1	:COMPUTE TOTAL SIZE OF CRB
	8F	30	018E	392	BSBW	IOGEN\$ALLOBLOCK	:ALLOCATE THE CRB
	FE6F	E8	0191	393	BLBS	R0,10\$	:BR IF SUCCESS
	03	31	0194	394	BRW	DB_ERROR	:ELSE - ERROR
	029A	88	0197	395	10\$: BISB	#LOAD_M_CRB,W*LOAD_FLAGS	:SET CRB LOADED FLAG
0000	CF	D0	019C	396	MOVL	R2,R9	:GET ADDR OF CRB
00000000	EF	D0	019F	397	MOVL	R9,ACF\$G_CRB	:*** SAVE FOR ACF\$UDA IN AUTOCONF
	59	B0	01A6	398	MOVW	R1,CRB\$W_SIZE(R9)	:SET SIZE
08	A9	90	01AA	399	MOVB	#DYN\$C_CRB,CRB\$B_TYPE(R9)	:SET TYPE
0A	A9	DE	01AE	400	MOVAL	CRB\$C_WQFL(R9),CRB\$C_WQFL(R9)	:SET WAIT QUEUE LISTHEAD
	69	DE	01B1	401	MOVAL	CRB\$C_WQFL(R9),CRB\$C_WQBL(R9)	
19	0D	E1	01B5	402	BBC	#DPT\$V_SUBCNTRL,DPT\$B_FLAGS(R11),CREATE_IDB	:BR IF NOT SUBCONTROLLER
	50	D0	01BA	403	MOVL	ACF\$C_ADAPTER(AP),R0	:GET ADDR OF ADAPTER CONTROL BLOCK
51	10	D0	01BD	404	MOVL	ADP\$C_CRB(R0),R1	:GET ADDR OF SECONDARY CRB

```

20 A9 51 D0 01C1 405      MOVL      R1,CRB$LINK(R9)      ;SET ADDR OF SECONDARY CRB
52 2C A1 D0 01C5 406      MOVL      CRB$INTD+VEC$IDB(R1),R2 ;GET ADDR OF SECONDARY IDB
53 0A AC 9A 01C9 407      MOVZBL    ACF$B_AUNIT(AP),R3    ;GET ADAPTER UNIT NUMBER
18 A243 25 A9 9E 01CD 408      MOVAB    CRB$INTD+1(R9),IDB$UCBLST(R2)[R3] ;SET ADDR OF DISPATCHER
                                01D3 409
                                01D3 410      : CREATE IDB
                                01D3 411
                                01D3 412      CREATE_IDB:
51 18 9A 01D3 413      MOVZBL    #IDB$K_LENGTH-<8*4>,R1    ;GET STANDARD LENGTH OF IDB
                                01D6 414      : (LESS NORMAL 8 UNITS)
54 16 AB 3C 01D6 415      MOVZWL    DPT$W_MAXUNITS(R11),R4  ;GET MAXIMUM NUMBER OF UNITS
1C AC B5 01DA 416      TSTW      ACF$W_MAXUNITS(AP)        ;WAS A VALUE SPECIFIED AT CONNECT?
04 13 01DD 417      BEQL      5$      ;BR IF NO
54 1C AC B0 01DF 418      MOVW      ACF$W_MAXUNITS(AP),R4    ;USE THE SPECIFIED VALUE INSTEAD
51 51 6144 DE 01E3 419 5$:      MOVAL    (R1)[R4],R1        ;COMPUTE TOTAL SIZE NEEDED
51 51 F7 01E7 420      CVTLW     R1,R1      ;IS MAXUNITS TOO LARGE ?
0A 1C 01EA 421      BVC      7$      ;BRANCH IF NO
50 007C80DA 8F D0 01EC 422      MOVL      #SYS$G$MAXTOOBIG,R0 ;SET ERROR
023B 31 01F3 423      BRW      DB_ERROR ;BRANCH TO ERROR
FE07 30 01F6 424 7$:      BSBW     IOGEN$ALLOBLOCK ;ALLOCATE THE IDB
03 50 E8 01F9 425      BLBS     R0,10$    ;BR IF SUCCESS
0232 31 01FC 426      BRW      DB_ERROR ;ELSE - ERROR
0000 CF 04 88 01FF 427 10$:    BISB     #LOAD_M_IDB,W^LOAD_FLAGS ;SET IDB LOADED FLAG
58 52 D0 0204 428      MOVL      R2,R8      ;GET ADDR OF IDB
08 A8 51 B0 0207 429      MOVW     R1,IDB$W_SIZE(R8) ;SET SIZE
0A A8 09 90 020B 430      MOVAB    #DYN$C_IDB,IDB$B_TYPE(R8) ;SET TYPE
0C A8 54 B0 020F 431      MOVW     R4,IDB$W_UNITS(R8) ;SET MAXIMUM UNITS
0C AB 05 91 0213 432      CMPB     #AT$_NULL,DPT$B_ADPTYPE(R11);NULL ADAPTER ?
15 12 0217 433      BNEQ     30$      ;IF NOT, BRANCH
2C A9 58 D0 0219 434      MOVL      R8,CRB$INTD+VEC$IDB(R9) ;STORE IDB POINTER IN CRB
38 A9 D4 021D 435      CLRL      CRB$INTD+VEC$ADP(R9) ;NULLIFY ADP POINTER
14 A8 D4 0220 436      CLRL      IDB$ADP(R8) ;NULLIFY ADP POINTER
68 D4 0223 437      CLRL      IDB$CSR(R8) ;NULLIFY CSR POINTER
FDDF CF D0 0225 438      MOVL      W^INT_DISP,- ;STORE INTERRUPT DISPATCHER CODE
24 A9 0229 439      CRB$INTD+VEC$Q_DISPATCH(R9)
009F 31 022B 440      BRW      CREATE_UCB ;CONTINUE
                                022E 441
                                022E 442 30$:
14 A8 6C D0 022E 443      MOVL      ACF$ADAPTER(AP),IDB$ADP(R8) ;SET ADDR OF ADP
68 0C AC D0 0232 444      MOVL      ACF$CONTRLREG(AP),IDB$CSR(R8) ;SET ADDR OF CSR
20 AC 90 0236 445      MOVAB    ACF$B_COMBO_CSR_OFFSET(AP),- ;SET OFFSET TO CSR START
0F A8 0239 446      IDB$B_COMBO_CSR_OFFSET(R8)
                                023B 447
                                023B 448      : Vector is now saved in IDB. The two least significant bits are always
                                023B 449      : zero, so the nine bit field is taken as bits 2 through 9 and placed
                                023B 450      : in a byte field. IDB$B_VECTOR is the field, and is currently not filled
                                023B 451      : in for a system booted from a UNIBUS boot device (e.g. an RK07).
                                023B 452
51 10 AC 07 02 EF 023B 453      extzv    #2,#7,acf$w_cvector(ap),r1 ; Get right shifted vector
0B A8 51 90 0241 454      movb     r1,idb$b_vector(r8) ; Place in IDB
                                0245 455
                                0245 456      MOVAB    ACF$B_COMBO_VECTOR_OFFSET(AP),- ;SET OFFSET TO VECTOR START
                                0248 457      IDB$B_COMBO_VECTOR_OFFSET(R8)
                                024A 458
                                024A 459      : CREATE INTERRUPT VECTOR DISPATCHER(S)
                                024A 460
                                024A 461

```

```

50 56 10 AC 32 024A 462 CREATE_VEC:
    0A 14 024A 463 CVTWL ACF$W_CVECTOR(AP),R6 ;GET VECTOR TABLE OFFSET
    007C802A 8F 14 024E 464 BGTR 5$ ;BR IF VECTOR SPECIFIED
    01D7 31 0250 465 MOVL #SYSG$ INVVEC,R0 ;SET INVALID VECTOR ERROR
    55 1E AC 9A 025A 466 BRW DB_ERROR ;...EXIT
    54 24 A9 DE 025E 467 5$: MOVZBL ACF$B_CNUMVEC(AP),R5 ;GET NUMBER OF INTERRUPT VECTORS
    64 FD9A CF D0 0262 468 MOVAL CRB$B_INTD(R9),R4 ;GET ADDR OF FIRST DISPATCHER
    0267 469 10$: MOVL W*MBINT_DISP,VEC$Q_DISPATCH(R4) ;ASSUME MBA DEVICE: SET
    0267 470 ; 'PUSHR #^M<R2,R3,R4,R5>'
    0267 471 ; AND 'JSB @#'
    14 A4 6C D0 0267 472 MOVL ACF$B_ADAPTER(AP),VEC$B_ADP(R4) ;SET ADDR OF ADP
    08 A4 58 D0 026B 473 MOVL R8,VEC$B_IDB(R4) ;SET ADDR OF IDB
    02 E1 026F 474 BBC #ACF$V_SCBVEC,- ;BR IF NOT VECTORING DIRECTLY
    13 0B AC 0271 475 ACF$B_AFLAG(AP),13$ ;FROM SCB
    00000000'EF C1 0274 476 ADDL3 MMG$A_SYSPARAM+<EXESGL_SCB-EXESA_SYSPARAM>,- ;USE SYS.EXE
    50 56 027A 477 R6,R0 ;COPY OF SCB ADDR TO GET ADDRESS
    027C 478 ; OF VECTOR IN SCB
    01 A4 9E 027C 479 MOVAB VEC$Q_DISPATCH+1(R4),- ;CONNECT DISPATCHER TO VECTOR
    60 027F 480 (R0) ;THE +1 MEANS USE THE INTERRUPT STACK
    64 FD84 CF D0 0280 481 MOVL W*INT_DISP,VEC$Q_DISPATCH(R4) ;STORE DISPATCHER CODE
    3D 11 0285 482 BRB 30$
    0287 483
    50 50 6C D0 0287 484 13$: MOVL ACF$B_ADAPTER(AP),R0 ;GET ADDR OF ADP
    56 10 A0 C1 028A 485 ADDL3 ADP$B_VECTOR(R0),R6,R0 ;GET ADDR OF VECTOR TABLE ENTRY
    0C AB 01 91 028F 486 #ATS_UBA,DPT$B_ADPTYPE(R1) ;UBA DEVICE?
    0C AB 0C 13 0293 487 BEQL 15$ ;BR IF YES
    0C AB 03 91 0295 488 CMPB #ATS_MPM,DPT$B_ADPTYPE(R1) ;MULTI-PORT MEMORY DEVICE?
    29 12 0299 489 BNEQ 30$ ;BR IF NOT
    60 02 A4 9E 029B 490 MOVAB VEC$Q_DISPATCH+2(R4),(R0) ;CONNECT DISPATCHER
    23 11 029F 491 BRB 30$
    02A1 492 15$:
    51 60 03 CB 02A1 493 BICL3 #3,(R0),R1 ;GET CURRENT VECTOR CONTENT
    00000000'8F 51 D1 02A5 494 ; EXCLUDING STACK SPECIFICATION
    13 13 02AC 495 CMPL R1,#UBA$UNEXINT ;IS VECTOR IN USE?
    00000000'8F 51 D1 02AE 496 BEQL 20$ ;BR IF NOT
    0A 13 02B5 497 CMPL R1,#CONSINTINP ;IS VECTOR IN USE BY THE CONSOLE TERMINAL DR
    50 007C801A 8F D0 02B7 498 BEQL 20$ ;IF EQL YES CONNECT THE TRUE DRIVER.
    0170 31 02BE 499 MOVL #SYSG$_VECINUSE,R0 ;SET ERROR STATUS
    02C1 500 BRW DB_ERROR ;...EXIT
    FD3C' 30 02C1 501 20$: BSBW IOGEN$CONN_VEC ;CONNECT UB DEVICE VECTOR
    56 04 C0 02C4 502 30$: ADDL #4,R6 ;INCREMENT VECTOR OFFSET
    54 24 C0 02C7 503 ADDL #VEC$K_LENGTH,R4 ;INCREMENT DISPATCH ADDR
    95 55 F5 02CA 504 SOBGTR R5,10$ ;DECREMENT VECTOR COUNT - BR IF MORE
    02CD 505
    02CD 506 ; CREATE A UCB
    02CD 507
    02CD 508
    02CD 509 CREATE_UCB:
    57 D4 02CD 510 CLRL R7 ;SET NO UCB
    51 0E AB 3C 02CF 511 MOVZWL DPT$W_UCBSIZE(R11),R1 ;GET SIZE OF UCB
    03 12 02D3 512 BNEQ 2$ ;BRANCH IF NON-NULL
    00E2 31 02D5 513 BRW INIT_DB ;BRANCH IF NO UCB FOR THIS DRIVER
    02D8 514
    57 51 D0 02D8 515 2$: MOVL R1,R7 ;COPY UCB SIZE FOR LATER
    00000058 8F C0 02DB 516 ADDL2 #ORB$C_LENGTH,R1 ;ALLOCATE ORB ADJACENT TO THE UCB
    FD1B' 30 02E2 517 BSBW IOGEN$ALLOBLOCK ;ALLOCATE THE UCB AND ORB
    03 50 E8 02E5 518 BLBS R0,5$ ;BR IF SUCCESS

```

0000	CF	08	31	02E8	519	BRW	DB_ERROR	:ELSE - ERROR
08	A2	57	88	02EB	520	5\$: BISB	#LOAD M_UCB,W^LOAD_FLAGS	:SET UCB LOADED FLAG
1C	A2	57	B0	02F0	521	MOVW	R7,UCB\$W_SIZE(R2)	:SET SIZE OF UCB
	57	52	C1	02F4	522	ADDL3	R7,R2,UCB\$L_ORB(R2)	:SET ORB ADDRESS
0A	A7	10	D0	02F9	523	MOVL	R2,R7	:COPY ADDR OF UCB
0C	A7	0C	90	02FC	524	MOVB	#DYN\$C_UCB,UCB\$B_TYPE(R7)	:SET TYPE
10	A7	0C	DE	0300	525	MOVAL	UCB\$L_ASTQFL(R7),UCB\$L_ASTQFL(R7)	:SET AST QUEUE LISTHEAD
24	A7	59	DE	0305	526	MOVAL	UCB\$L_ASTQFL(R7),UCB\$L_ASTQBL(R7)	:...
	0C	A9	D0	030A	527	MOVL	R9,UCB\$L_CRB(R7)	:SET ADDR OF CRB
50	12	AC	B6	030E	528	INCW	CRB\$W_REFC(R9)	:INCREMENT CRB REFERENCE COUNT
0C	A8	50	3C	0311	529	MOVZWL	ACF\$W_CUNIT(AP),R0	:GET CONTROLLER UNIT NUMBER
		0A	B1	0315	530	CMPW	R0,IDB\$W_UNITS(R8)	:MAXIMUM UNITS EXCEEDED?
50	007C803A	8F	1F	0319	531	BLSSU	8\$	:BR IF NOT
	010C	31	D0	031B	532	MOVL	#SYS\$G_MAXUNITS,R0	:SET MAXIMUM UNITS EXCEEDED
			31	0322	533	BRW	DB_ERROR	:...EXIT
				0325	534	8\$:		
18	A840	57	D0	0325	535	MOVL	R7,IDB\$L_UCBLST(R8)[R0]	:SET ADDR OF UCB IN IDB
28	A7	5A	D0	032A	536	MOVL	R10,UCB\$L_DDB(R7)	:SET ADDR OF DDB
4C	A7	4C	DE	032E	537	MOVAL	UCB\$L_IOQFL(R7),UCB\$L_IOQFL(R7)	:SET I/O QUEUE LISTHEAD
50	A7	4C	DE	0333	538	MOVAL	UCB\$L_IOQFL(R7),UCB\$L_IOQBL(R7)	:...
54	A7	12	B0	0338	539	MOVW	ACF\$W_CUNIT(AP),UCB\$W_UNIT(R7)	:SET UNIT NUMBER
	51	2C	D0	033D	540	MOVL	CRB\$L_INTD+VEC\$L_IDB(R9),R1	:IDB ADDRESS
	51	14	D0	0341	541	MOVL	IDB\$L_ADP(R1),R1	:ADP ADDRESS
		0C	13	0345	542	BEQL	9\$	:PHYSICAL DEVICE IF NON-ZERO
0E	A1	00	B1	0347	543	CMPW	#AT\$MBA,ADP\$W_ADPTYPE(R1)	:IS IT A MASSBUS
		06	12	034B	544	BNEQ	9\$	:BRANCH IF NOT
0090	C7	0A	90	034D	545	MOVB	ACF\$B_AUNIT(AP),UCB\$B_SLAVE(R7)	:SET SLAVE CNTRLER NUMBER
				0353	546			
				0353	547			: BEFORE HOOKING THE UCB UP TO THE DDB CHAIN, INITIALIZE THE ORB.
				0353	548			
51	1C	A7	D0	0353	549	9\$: MOVL	UCB\$L_ORB(R7),R1	:GET THE ORB ADDRESS
0A	A1	49	90	0357	550	MOVB	#DYN\$C_ORB,ORB\$B_TYPE(R1)	:SET BLOCK TYPE
08	A1	0058	B0	035C	551	MOVW	#ORB\$C_LENGTH,ORB\$W_SIZE(R1)	:SET BLOCK SIZE
	0B	A1	90	0362	552	MOVB	#ORB\$M_PROT_16,ORB\$B_FLAGS(R1)	:SOGW PROT WORD & NO ACL QUEUE
				0366	553			
				0366	554	ASSUME	ORB\$L_ACL_DESC EQ ORB\$L_ACL_COUNT+4	
				0366	555			
28	A1	7C		0366	556	CLRQ	ORB\$L_ACL_COUNT(R1)	:ACL IS EMPTY
				0369	557			
				0369	558			: NOW HOOK THE UCB INTO THE DDB CHAIN
				0369	559			
51	04	AA	DE	0369	560	MOVAL	DDB\$L_UCB(R10),R1	:GET ADDR OF ADDR OF FIRST UCB
50	61		D0	036D	561	MOVL	(R1),R0	:GET ADDR OF FIRST UCB
	09		13	0370	562	BEQL	20\$	:BR IF NONE
51	30	A0	DE	0372	563	10\$: MOVAL	UCB\$L_LINK(R0),R1	:GET ADDR OF ADDR OF NEXT UCB
50	61		D0	0376	564	MOVL	(R1),R0	:GET ADDR OF NEXT UCB
	F7		12	0379	565	BNEQ	10\$	:BR IF IT EXISTS
0005	CF	51	D0	037B	566	20\$: MOVL	R1,W^UCB_BLINK	:SAVE UCB BACKWARD LINK
	61	57	D0	0380	567	MOVL	R7,(R1)	:SET FORWARD LINK TO UCB
				0383	568			
32	0D	AB	E1	0383	569	BBC	#DPT\$V_SVP,DPT\$B_FLAGS(R11),INIT DB	:BR IF NO SYS PAGE REQUIRED
52	00000000	EF	9E	0388	570	MOVAB	MMG\$A_SYSPARAM,R2	:GET ADDR OF SYS PARAMS
50	0000	C2	D0	038F	571	MOVL	BOO\$G_SPTFREL-EXESA_SYSPARAM(R2),R0	:GET NEXT FREE SPT ENTRY
	0000	C2	D1	0394	572	CMPL	R0,BOO\$G_SPTFREL-EXESA_SYSPARAM(R2)	:SPT FULL?
		0A	1B	0399	573	BLEQU	30\$	:BR IF NO
50	007C8022	8F	D0	039B	574	MOVL	#SYS\$G_SPTFULL,R0	:SET ERROR STATUS
	008C	31		03A2	575	BRW	DB_ERROR	:...EXIT

```

6140 51 0000'C2 D6 03A5 576 30$: INCL BOO$GL_SPTFREL-EXESA_SYSPARAM(R2) ;INCREMENT NEXT FREE POINTER
      0000'C2 D0 03A9 577      MOVL MMG$GL_SPTBASE-EXESA_SYSPARAM(R2),R1 ;GET ADDR OF SPT
      901FFFFF 8F D0 03AE 578      MOVL #<PTESM_VALID!PTESC_RW!PTESM_PFN>,(R1)[R0] ;VALIDATE THIS ENTRY
      74 A7 50 D0 03B6 579      MOVL R0,UCB$C_SVPN(R7) ;SET-SYS PAGE NUMBER IN UCB
      03BA 580
      03BA 581 : INITIALIZE ALL THE CONTROL BLOCKS
      03BA 582
      03BA 583 INIT_DB:
      54 5B D0 03BA 584      MOVL R11,R4 ;SET ADDR OF DRIVER PROLOGUE
      55 57 D0 03BD 585      MOVL R7,R5 ;SET ADDR OF UCB
      00000000'GF 16 03C0 586      JSB G^IOC$INITDRV ;INIT THE CONTROL BLOCKS
      OC AA D5 03C6 587      TSTL DDB$C_DDT(R10) ;CHECK IF DDT EXISTS
      03 13 03C9 588      BEQL 10$ ;IF EQL, BAD DDB
      OA 50 E8 03CB 589      BLBS R0,70$ ;BR IF SUCCESS
      50 007C800A 8F D0 03CE 590 10$: MOVL #SYSG$ INVDPNTINI,R0 ;SET ERROR STATUS
      0059 31 03D5 591      BRW DB_ERROR ;...EXIT
      03D8 592
      03D8 593 : Relocate the DDT and FDT to SYSTEM VIRTUAL ADDRESSES.
      03D8 594
      5B DD 03D8 595 70$: PUSHL R11 ;SAVE DPT ADDRESS
      5B 5A D0 03DA 596      MOVL R10,R11 ;SET ADDRESS OF DDB
      00000000'GF 16 03DD 597      JSB G^IOC$RELOC_DDT ;RELOCATE DDT and FDT
      5B 8ED0 03E3 598      POPL R11 ;RESTORE DPT
      03E6 599
      03E6 600 : CALL THE CONTROLLER AND DRIVE INITIALIZATION ROUTINES
      03E6 601
      03E6 602 INIT_CNTRL:
      03E6 603
      50 01 D0 03E6 604      DSBINT ;DISABLE INTERRUPTS
      03EC 605      MOVL #1,R0 ;INSURE CSR TEST SEE SUCCESS EVEN IF
      03EF 606      ;IOGEN$CNTRL_INI IS NOT CALLED
      06 0000'CF 01 E1 03EF 607      BBC #LOAD_V_CRB,W^LOAD_FLAGS,INIT_UNIT ;BR IF CRB NOT JUST CREATED
      56 5A D0 03F5 608      MOVL R10,R6 ;SET ADDR OF DDB
      FC05' 30 03F8 609      BSBW IOGEN$CNTRL_INI ;INITIALIZE THE CONTROLLER
      03FB 610      ;***BEWARE*** DON'T CORRUPT R0
      03FB 611      ;BEFORE THE BLBC BELOW
      03FB 612
      03FB 613 INIT_UNIT:
      03FB 614
      55 57 D0 03FB 615      MOVL R7,R5 ;SETUP UCB ADDRESS
      17 13 03FE 616      BEQL ENABLE_INT ;BRANCH IF NO UCB TO INIT
      0088 C5 OC AA D0 0400 617      MOVL DDB$C_DDT(R10), - ;COPY DDT ADDRESS TO UCB
      0406 618      UCB$C_DDT(R5)
      OE 50 E9 0406 619      BLBC R0,ENABLE_INT ;BRANCH IF ERROR TESTING CSR
      0409 620      ; (RETURNED BY IOGEN$CNTRL_INI ABOVE)
      58 DD 0409 621      PUSHL R8 ;SAVE IDB ADDRESS
      58 59 D0 040B 622      MOVL R9,R8 ;SETUP CRB ADDRESS
      00000000'GF 16 040E 623      JSB G^IOC$UNITINIT ;CALL COMMON DRIVER UNIT INIT CALLER
      58 8ED0 0414 624      POPL R8 ;RESTORE IDB ADDRESS
      0417 625
      0417 626 ENABLE_INT:
      0417 627      ENBINT ;RE-ENABLE INTERRUPTS
      041A 628
      041A 629 NEXT_UCB:
      041A 630
      041A 631 : Increment current unit number
      041A 632 : Loop through Create UCB code once for each unit

```

```

21 AC 95 041A 633 ;
   OB 13 041A 634 TSTB ACF$B_NUMUNIT(AP) ; TEST NO. OF UNITS TO CONFIGURE
21 AC 97 041D 635 BEQL DB_EXIT ; EXIT IF ZERO
   O6 15 041F 636 DECB ACF$B_NUMUNIT(AP) ; DECREMENT NO. OF UNITS TO CONFIGURE
12 AC B6 0422 637 BLEQ DB_EXIT ; ALL DONE!
FEA3 31 0424 638 INCW ACF$W_CUNIT(AP) ; MOVE TO NEXT UNIT NO.
      0427 639 BRW CREATE_UCB ; AND LOOP
      042A 640
      042A 641
      042A 642 : OPERATION COMPLETED - UNLOCK I/O DATABASE AND EXIT
      042A 643
      042A 644 DB_EXIT:
FBD3' 30 042A 645 BSBW IOGEN$UNLK_IODB ;UNLOCK I/O DATABASE
50 01 D0 042D 646 ; AND LOWER IPL
      04 042D 647 MOVL #1,R0 ;SET SUCCESS
      0430 648 RET ;...EXIT
      0431 649

```

```

0431 651 .SBTTL DB_ERROR - ERROR LOADING DATABASE
0431 652 :++
0431 653 :
0431 654 DB_ERROR - LOCAL ROUTINE TO UNLOAD A PARTIALLY LOADED DATABASE
0431 655 :
0431 656 This routine checks the loading flags (LOAD_FLAGS) and
0431 657 unload and unlinks any control blocks that were just created.
0431 658 :
0431 659 INPUTS:
0431 660 :
0431 661 LOAD_FLAGS = FLAGS INDICATING CONTROL BLOCKS THAT HAVE JUST
0431 662 BEEN CREATED
0431 663 :
0431 664 OUTPUTS:
0431 665 :
0431 666 R0 = ERROR STATUS THAT CAUSED UNLOADING
0431 667 :
0431 668 :--
0431 669 DB_ERROR:
50 DD 0431 670 PUSHL R0 ;SAVE ERROR STATUS
0433 671 :
0433 672 UCB UNLOADING
0433 673 :
26 0000'CF 03 E1 0433 674 BBC #LOAD_V_UCB,W^LOAD_FLAGS,20$ ;BR IF UCB NOT CREATED
50 007C803A 8F D1 0439 675 CMPL #SYSGB_MAXUNITS,R0 ;WAS IT A MAXUNITS ERROR?
04 13 0440 676 BEQL 5$ ;IF SO, NO BLINK
0005'DF D4 0442 677 CLRL @W^UCB.BLINK ;CLEAR POINTER TO UCB
0C A9 B7 0446 678 5$: DECW CRB$W_REFC(R9) ;DECREMENT CRB REFERENCE COUNT
50 12 AC 3C 0449 679 MOVZWL ACFSW_CUNIT(AP),R0 ;GET CONTROLLER UNIT NUMBER
18 A840 D4 044D 680 CLRL IDB$L_UCBLST(R8)[R0] ;CLEAR IDB POINTER TO UCB
50 57 D0 0451 681 MOVL R7,R0 ;SET ADDR OF UCB
02 10 0454 682 BSBB 10$ ;DEALLOCATE THE UCB
07 11 0456 683 BRB 20$ ;
0458 684 :
0458 685 LOCAL SUBROUTINE TO DEALLOCATE A CONTROL BLOCK
0458 686 :
0458 687 INPUT - R0 = ADDRESS OF BLOCK TO DEALLOCATE
0458 688 :
00000000'GF 16 0458 689 10$: JSB G^EXE$DEANONPAGED ;DEALLOCATE TO NON-PAGED POOL
05 05 045E 690 RSB
045F 691 :
045F 692 :
045F 693 IDB UNLOADING
045F 694 :
05 0000'CF 02 E1 045F 695 20$: BBC #LOAD_V_IDB,W^LOAD_FLAGS,30$ ;BR IF IDB NOT CREATED
50 58 D0 0465 696 MOVL R8,R0 ;SET ADDR OF IDB
EE 10 0468 697 BSBB 10$ ;DEALLOCATE THE IDB
046A 698 :
046A 699 CRB UNLOADING
046A 700 :
1A 0000'CF 01 E1 046A 701 30$: BBC #LOAD_V_CRB,W^LOAD_FLAGS,50$ ;BR IF CRB NOT CREATED
10 0D AB 00 E1 0470 702 BBC #DPT$V_SUBCNTRL,DPT$B_FLAGS(R11),40$ ;BR IF NOT SUBCONTROLLER
51 20 A9 D0 0475 703 MOVL CRB$L_CINK(R9),R1 ;GET ADDR OF SECONDARY CRB
51 2C A1 D0 0479 704 MOVL CRB$L_INTD+VEC$L_IDB(R1),R1 ;GET ADDR OF SECONDARY IDB
50 0A AC 9A 047D 705 MOVZBL ACFSB_AUNIT(AP),R0 ;GET ADAPTER UNIT NUMBER
18 A140 D4 0481 706 CLRL IDB$L_UCBLST(R1)[R0] ;CLEAR DISPATCHER POINTER
50 59 D0 0485 707 40$: MOVL R9,R0 ;SET ADDR OF CRB

```

```

CE 10 0488 708 BSBB 10$ ;DEALLOCATE THE CRB
      048A 709 :
      048A 710 : DDB UNLOADING
      048A 711 :
OD 0000'CF 00 E1 048A 712 50$: BBC #LOAD_V DDB,W^LOAD_FLAGS,60$ :BR IF DDB NOT CREATED
      OB AB 97 0490 713 DECB DPT$B_REFC(R11) ;DECREMENT DPT REFERENCE COUNT
      0001'DF 6A D0 0493 714 MOVL DDB$L_LINK(R10),@W^DDB_BLINK;UPDATE THE DDB CHAIN
      50 5A D0 0498 715 MOVL R10,R0 ;SET ADDR OF DDB
      BB 10 049B 716 BSBB 10$ ;DEALLOCATE THE DDB
      049D 717 60$:
FB60' 30 049D 718 BSBW IOGEN$UNLK_IODB ;UNLOCK THE I/O DATABASE
      50 8ED0 04A0 719 ; AND LOWER IPL
      04 04A3 720 POPL R0 ;RESTORE STATUS
      04A4 721 RET ;...EXIT
      04A4 722
      04A4 723 .END

```

LOADER
Symbol table

E 15
- LOAD A DRIVER AND/OR DEVICE CONTROL BL 15-SEP-1984 23:54:17 VAX/VMS Macro V04-00
4-SEP-1984 23:04:34 [BOOTS.SRC]LOADER.MAR;1

Page 17
(1)

ACFSB_AFLAG	= 0000000B		
ACFSB_AUNIT	= 0000000A		
ACFSB_CNUMVEC	= 0000001E		
ACFSB_COMBO_CSR_OFFSET	= 00000020		
ACFSB_COMBO_VECTOR_OFFSET	= 0000001F		
ACFSB_NUMUNIT	= 00000021		
ACF\$GL_CRB	*****	X	03
ACF\$GL_DDB	*****	X	03
ACF\$GL_DPT	*****	X	03
ACF\$GL_IDB	*****	X	03
ACF\$GL_LASTDDB	*****	X	03
ACF\$GL_SB	*****	X	03
ACF\$GL_UCB	*****	X	03
ACF\$SL_ADAPTER	= 00000000		
ACF\$SL_CONTRLREG	= 0000000C		
ACF\$SL_DEVNAME	= 00000014		
ACF\$SL_DRVNAME	= 00000018		
ACF\$V_CRBBLT	= 00000001		
ACF\$V_GETDONE	= 00000005		
ACF\$V_NOLOAD_DB	= 00000003		
ACF\$V_RELOAD	= 00000000		
ACF\$V_SCBVEC	= 00000002		
ACF\$V_SUPPORT	= 00000004		
ACF\$W_CUNIT	= 00000012		
ACF\$W_CVECTOR	= 00000010		
ACF\$W_MAXUNITS	= 0000001C		
ADP\$SL_CRB	= 00000010		
ADP\$SL_VECTOR	= 00000010		
ADP\$W_ADPTYPE	= 0000000E		
ATS_CI	= 00000004		
ATS_DR	= 00000002		
ATS_MBA	= 00000000		
ATS_MPM	= 00000003		
ATS_NULL	= 00000005		
ATS_UBA	= 00000001		
BIT...	= 00000004		
BOO\$GL_SPTFREL	*****	X	03
BOO\$GL_SPTFREL	*****	X	03
CHECK_DRV	0000005F	R	03
CLUS\$GL_ALLOCLS	*****	X	03
CONSINTINP	*****	X	03
CRB\$B_TYPE	= 0000000A		
CRB\$K_LENGTH	= 00000048		
CRB\$SL_INTD	= 00000024		
CRB\$SL_LINK	= 00000020		
CRB\$SL_WQBL	= 00000004		
CRB\$SL_WQFL	= 00000000		
CRB\$W_REFC	= 0000000C		
CRB\$W_SIZE	= 00000008		
CREATE_CRB	00000180	R	03
CREATE_DDB	000000F6	R	03
CREATE_IDB	000001D3	R	03
CREATE_UCB	000002CD	R	03
CREATE_VEC	0000024A	R	03
DB_ERROR	00000431	R	03
DB_EXIT	0000042A	R	03
DDB\$B_TYPE	= 0000000A		

DDB\$K_LENGTH	= 00000044		
DDB\$SL_ALLOCLS	= 0000003C		
DDB\$SL_DDT	= 0000000C		
DDB\$SL_LINK	= 00000000		
DDB\$SL_SB	= 00000034		
DDB\$SL_UCB	= 00000004		
DDB\$T_DRVNAME	= 00000024		
DDB\$T_NAME	= 00000014		
DDB\$W_SIZE	= 00000008		
DDB BCINK	00000001	R	02
DPT\$B_ADPTYPE	= 0000000C		
DPT\$B_FLAGS	= 0000000D		
DPT\$B_REFC	= 0000000B		
DPT\$SL_FLINK	= 00000000		
DPT\$T_NAME	= 00000020		
DPT\$V_SUBCNTRL	= 00000000		
DPT\$V_SVP	= 00000001		
DPT\$W_MAXUNITS	= 00000016		
DPT\$W_UCBSIZE	= 0000000E		
DYN\$C_CRB	= 00000005		
DYN\$C_DDB	= 00000006		
DYN\$C_IDB	= 00000009		
DYN\$C_ORB	= 00000049		
DYN\$C_UCB	= 00000010		
ENABLE_INT	00000417	R	03
EXESA_SYSPARAM	*****	X	03
EXES\$DEANONPAGED	*****	X	03
EXES\$GL_SCB	*****	X	03
IDB\$B_COMBO_CSR_OFFSET	= 0000000F		
IDB\$B_COMBO_VECTOR_OFFSET	= 00000010		
IDB\$B_TYPE	= 0000000A		
IDB\$B_VECTOR	= 0000000B		
IDB\$K_LENGTH	= 00000038		
IDB\$SL_ADP	= 00000014		
IDB\$SL_CSR	= 00000000		
IDB\$SL_UCBLST	= 00000018		
IDB\$W_SIZE	= 00000008		
IDB\$W_UNITS	= 0000000C		
INIT_CNTRL	000003E6	R	03
INIT_DB	000003BA	R	03
INIT_UNIT	000003FB	R	03
INT_DISP	00000008	R	03
IOCS\$GL_DPTLIST	*****	X	03
IOCS\$INITDRV	*****	X	03
IOCS\$RELOC_DDT	*****	X	03
IOCS\$UNITINIT	*****	X	03
IOGEN\$ALLOBLOCK	*****	X	03
IOGEN\$CNTRL_INI	*****	X	03
IOGEN\$CONN_VEC	*****	X	03
IOGEN\$LOADDRIV	*****	X	03
IOGEN\$LOADER	00000010	RG	03
IOGEN\$LOCK_IDB	*****	X	03
IOGEN\$UNLK_IDB	*****	X	03
IPL\$ SYNCH	= 00000008		
LOAD_DB	0000009F	R	03
LOAD_FLAGS	00000000	R	02
LOAD_M_CRB	= 00000002		

LOADER
Symbol table

F 15

- LOAD A DRIVER AND/OR DEVICE CONTROL BL 15-SEP-1984 23:54:17 VAX/VMS Macro V04-00
4-SEP-1984 23:04:34 [BOOTS.SRC]LOADER.MAR;1

Page 18
(1)

LOAD_M_DDB	= 00000001		
LOAD_M_IDB	= 00000004		
LOAD_M_UCB	= 00000008		
LOAD_V_CRB	= 00000001		
LOAD_V_DDB	= 00000000		
LOAD_V_IDB	= 00000002		
LOAD_V_UCB	= 00000003		
MBINT_DISP	00000000	R	03
MMGSA_SYSPARAM	*****	X	03
MMGSGC_SPTBASE	*****	X	03
NEXT_UCB	0000041A	R	03
ORBSB_FLAGS	= 0000000B		
ORBSB_TYPE	= 0000000A		
ORBSL_LENGTH	= 00000058		
ORBSL_ACL_COUNT	= 00000028		
ORBSL_ACL_DESC	= 0000002C		
ORBSM_PROT_16	= 00000001		
ORBSW_SIZE	= 00000008		
PRS_IPL	= 00000012		
PTESC_KW	= 10000000		
PTESM_PFN	= 001FFFFFF		
PTESM_VALID	= 80000000		
SGNSGET_DEVICE	*****	X	03
SIZ...	= 00000001		
SS\$NOPRIV	*****	X	03
SYSCMKRNL	*****	GX	03
SYSG\$INVDPTINI	= 007C800A		
SYSG\$INVVEC	= 007C802A		
SYSG\$MAXTOOBIG	= 007C80DA		
SYSG\$MAXUNITS	= 007C803A		
SYSG\$SPTFULL	= 007C8022		
SYSG\$VECIUSE	= 007C801A		
UBASUNEXINT	*****	X	03
UCBSB_SLAVE	= 00000090		
UCBSB_TYPE	= 0000000A		
UCBSL_ASTQBL	= 00000010		
UCBSL_ASTQFL	= 0000000C		
UCBSL_CRB	= 00000024		
UCBSL_DDB	= 00000028		
UCBSL_DDT	= 00000088		
UCBSL_IOQBL	= 00000050		
UCBSL_IOQFL	= 0000004C		
UCBSL_LINK	= 00000030		
UCBSL_ORB	= 0000001C		
UCBSL_SVPN	= 00000074		
UCBSW_SIZE	= 00000008		
UCBSW_UNIT	= 00000054		
UCB_BLINK	00000005	R	02
VECSK_LENGTH	= 00000024		
VECSL_ADP	= 00000014		
VECSL_IDB	= 00000008		
VECSQ_DISPATCH	= 00000000		

+-----+
! 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
NONPAGED_DATA	00000009 (9.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC QUAD
NONPAGED_CODE	000004A4 (1188.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	39	00:00:00.06	00:00:00.38
Command processing	155	00:00:00.70	00:00:02.45
Pass 1	366	00:00:13.40	00:00:25.34
Symbol table sort	0	00:00:01.93	00:00:03.75
Pass 2	140	00:00:02.92	00:00:05.58
Symbol table output	20	00:00:00.15	00:00:00.37
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	724	00:00:19.18	00:00:37.89

The working set limit was 1650 pages.
77352 bytes (152 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1327 non-local and 42 local symbols.
723 source lines were read in Pass 1, producing 23 object records in Pass 2.
31 pages of virtual memory were used to define 30 macros.

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

Macro library name	Macros defined
\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	18
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	27

1446 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:LOADER/OBJ=OBJ\$:LOADER MSRC\$:LOADER/UPDATE=(ENH\$:LOADER)+EXECMLS/LIB+LIB\$:BOOTS.MLB/LIB

0038 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY