

MM		MM		AAAAAA		KK		KK	RRRRRRRR		VV		VV	TTTTTTTTTT	
MM		MM		AAAAAA		KK		KK	RRRRRRRR		VV		VV	TTTTTTTTTT	
MMM	MMM	AA	AA			KK		KK	RR	RR	VV	VV		TT	
MMM	MMM	AA	AA			KK		KK	RR	RR	VV	VV	VV	TT	
MM	MM	MM	AA	AA		KK	KK	RR	RR	RR	VV	VV	VV	TT	
MM	MM	MM	AA	AA		KK	KK	RR	RR	RR	VV	VV	VV	TT	
MM		MM	AA	AA	KKKKKK			RRRRRRRR			VV	VV	VV	TT	
MM		MM	AA	AA	KKKKKK			RRRRRRRR			VV	VV	VV	TT	
MM		MM	AAAAAAAAA		KK	KK		RR	RR		VV	VV	VV	TT	
MM		MM	AAAAAAAAA		KK	KK		RR	RR		VV	VV	VV	TT	
MM		MM	AA	AA	KK		KK	RR	RR	RR	VV	VV	VV	TT	
MM		MM	AA	AA	KK		KK	RR	RR		VV	VV	VV	TT	
MM		MM	AA	AA	KK		KK	RR	RR	RR		VV	VV	TT	
MM		MM	AA	AA	KK		KK	RR	RR			VV	VV	TT	
MM		MM	AA	AA	KK		KK	RR	RR			VV	VV	TT	

```

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
LL          II             SSSSSS
LLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLL IIIIII          SSSSSSSS

```



```
1 0001 0 MODULE MAKRVT (
2 0002 0
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 IDENT = 'V04-000'
5 0005 1 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: MOUNT Utility Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This module creates the relative volume table for a disk volume set.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 23-Oct-1978 11:12
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-003 HH0041 Hai Huang 24-Jul-1984
52 0052 1 Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
53 0053 1
54 0054 1 V03-002 CDS0002 Christian D. Saether 22-Aug-1983
55 0055 1 Teach I/O database search about system blocks.
56 0056 1
57 0057 1 V03-001 CDS0001 Christian D. Saether 2-Aug-1983
```

MAKRV
V04-000

M 11
16-Sep-1984 01:17:34
14-Sep-1984 12:45:22

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]MAKRV.B32;1 Page 2 (1)

```

: 58      0058 1 | Remove reference to obsolete RVX structure.
: 59      0059 1 |
: 60      0060 1 | V02-003 ACG0229      Andrew C. Goldstein,    4-Jan-1982  15:04
: 61      0061 1 |      Add volume set name to RVT
: 62      0062 1 |
: 63      0063 1 | V02-002 ACG0178      Andrew C. Goldstein,    9-Jun-1980  13:35
: 64      0064 1 |      Skip magtapes and foreign volumes in mounted volume set search
: 65      0065 1 |
: 66      0066 1 | V02-001 ACG0175      Andrew C. Goldstein,    23-May-1980  14:44
: 67      0067 1 |      Add I/O database lock cleanup, tie off RVT extension
: 68      0068 1 |
: 69      0069 1 | V02-000 ACG0167      Andrew C. Goldstein,    18-Apr-1980  13:38
: 70      0070 1 |      Previous revision history moved to MOUNT.REV
: 71      0071 1 |      **
: 72      0072 1 |
: 73      0073 1 |
: 74      0074 1 | LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 75      0075 1 | REQUIRE 'SRC$:MOUDEF.B32';
```

```

: 77 0607 1 GLOBAL ROUTINE ENTER_RVT (NAME, NEW_UCB) =
: 78 0608 1
: 79 0609 1 ++
: 80 0610 1
: 81 0611 1 FUNCTIONAL DESCRIPTION:
: 82 0612 1
: 83 0613 1 This routine finds the relative volume table of the volume set of this
: 84 0614 1 volume, if it exists, and enters the volume in the table. If the RVT
: 85 0615 1 does not exist, it creates it. This routine also does the initial
: 86 0616 1 binding of a new volume into a set. This routine must be called in kernel
: 87 0617 1 mode.
: 88 0618 1
: 89 0619 1
: 90 0620 1 CALLING SEQUENCE:
: 91 0621 1 ENTER_RVT (ARG1, ARG2)
: 92 0622 1
: 93 0623 1 INPUT PARAMETERS:
: 94 0624 1 ARG1: address of descriptor of volume set name
: 95 0625 1 ARG2: UCB address of volume being mounted
: 96 0626 1
: 97 0627 1 IMPLICIT INPUTS:
: 98 0628 1 HOME_BLOCK: home block of volume being mounted
: 99 0629 1
: 100 0630 1 OUTPUT PARAMETERS:
: 101 0631 1 NONE
: 102 0632 1
: 103 0633 1 IMPLICIT OUTPUTS:
: 104 0634 1 REAL_RVT: address of RVT found or created
: 105 0635 1
: 106 0636 1 ROUTINE VALUE:
: 107 0637 1 address of RVT
: 108 0638 1
: 109 0639 1 SIDE EFFECTS:
: 110 0640 1 RVT may be created or modified
: 111 0641 1 HOME_BLOCK altered if /BIND performed
: 112 0642 1
: 113 0643 1 --
: 114 0644 1
: 115 0645 2 BEGIN
: 116 0646 2
: 117 0647 2 MAP
: 118 0648 2 NAME : REF VECTOR; ! volume name string descriptor
: 119 0649 2
: 120 0650 2 LABEL
: 121 0651 2 SEARCH_LOOP; ! outer device search loop
: 122 0652 2
: 123 0653 2 LOCAL
: 124 0654 2 ENTRY_COUNT, ! number of entries needed in RVT
: 125 0655 2 SIZE, ! size of new RVT
: 126 0656 2 DDB : REF BBLOCK, ! pointer to current DDB
: 127 0657 2 SB : REF BBLOCK, ! system block pointer
: 128 0658 2 UCB : REF BBLOCK, ! pointer to current UCB
: 129 0659 2 VCB : REF BBLOCK, ! pointer to current VCB
: 130 0660 2 RVT : REF BBLOCK, ! pointer to current RVT
: 131 0661 2 NEW_RVT : REF BBLOCK, ! pointer to newly created RVT
: 132 0662 2 FCB : REF BBLOCK, ! pointer to current FCB
: 133 0663 2 WINDOW : REF BBLOCK; ! pointer to current window
```



```
134 0664 2  
135 0665 2 EXTERNAL  
136 0666 2 MOUNT_OPTIONS : BITVECTOR, ! MOUNT command options  
137 0667 2 CLEANUP_FLAGS : BITVECTOR, ! cleanup action flags  
138 0668 2 REAL_VCB : REF BBLOCK, ! address of VCB allocated  
139 0669 2 REAL_RVT : REF BBLOCK, ! address of newly allocated RVT  
140 0670 2 HOME_BLOCK : BBLOCK, ! volume home block buffer  
141 0671 2 SCH$GL_CURPCB : REF BBLOCK ADDRESSING_MODE (ABSOLUTE),  
142 0672 2 ! address of process PCB  
143 0673 2 SC$SGQ_CONFIG : ADDRESSING_MODE (ABSOLUTE);  
144 0674 2 ! I/O database listhead  
145 0675 2  
146 0676 2 EXTERNAL ROUTINE  
147 0677 2 LOCK_IODB, ! lock I/O database mutex  
148 0678 2 UNLOCK_IODB, ! unlock I/O database mutex  
149 0679 2 ALLOCATE_MEM, ! allocate dynamic memory  
150 0680 2 DEALLOCATE_MEM; ! deallocate dynamic memory  
151 0681 2  
152 0682 2  
153 0683 2 ! Needless to say, the search must be done with the I/O database locked to  
154 0684 2 ! prevent list perturbations. We run down the DDB list, following the UCB  
155 0685 2 ! list off each one, looking for file structured devices that are mounted  
156 0686 2 ! but not allocated.  
157 0687 2  
158 0688 2  
159 0689 2 LOCK_IODB ();  
160 0690 2 CLEANUP_FLAGS[CLF_UNLOCKDB] = 1;  
161 0691 2  
162 0692 2 SB = .SC$SGQ_CONFIG;  
163 0693 2  
164 0694 2 SEARCH_LOOP:  
165 0695 2 BEGIN  
166 0696 2  
167 0697 2 UNTIL .SB EQLA SC$SGQ_CONFIG  
168 0698 2 DO  
169 0699 2 BEGIN  
170 0700 2  
171 0701 2 DDB = .SB [SB$L_DDB];  
172 0702 2  
173 0703 2 WHILE .DDB NEQ 0  
174 0704 2 DO  
175 0705 2 BEGIN  
176 0706 2 IF .DDB[DDB$B_TYPE] NEQ DYN$C_DDB  
177 0707 2 THEN BUG_CHECK (NOTDDBDDB, FATAL, 'Corrupted DDB list');  
178 0708 2 UCB = .DDB[DDB$L_UCB];  
179 0709 2  
180 0710 2 UNTIL .UCB EQL 0 DO  
181 0711 2 BEGIN  
182 0712 2 IF .UCB[UCB$B_TYPE] NEQ DYN$C_UCB  
183 0713 2 THEN BUG_CHECK (NOTUCBUCEB, FATAL, 'Corrupted UCB list');  
184 0714 2  
185 0715 2 IF .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_FOD]  
186 0716 2 AND .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_MNT]  
187 0717 2 AND NOT .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_FOR]  
188 0718 2 AND NOT .BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_SQD]  
189 0719 2 AND (IF .MOUNT_OPTIONS[OPT_NOSHARE]  
190 0720 2 THEN (.BBLOCK[UCB[UCB$L_DEVCHAR], DEV$V_ALL]
```

```
191      AND .UCB[UCB$$_PID] EQL .SCH$GL_CURPCB[PCB$$_PID])
192      ELSE NOT .BBLOCK[UCB[UCB$$_DEVCHAR], DEV$V_ALL]
193      )
194      THEN
195      BEGIN
196      VCB = .UCB[UCB$$_VCB];
197      IF .VCB[VCB$$_TYPE] NEQ DYN$C_VCB
198      THEN BUG_CHECK (NOTVCBUCB, FATAL, 'Not VCB pointer in UCB');
199
200      RVT = .VCB[VCB$$_RVT];
201      IF .RVT NEQ 0
202      THEN IF .RVT[RVT$$_TYPE] NEQ DYN$C_UCB
203      THEN
204      BEGIN
205      IF .RVT[RVT$$_TYPE] NEQ DYN$C_RVT
206      THEN BUG_CHECK (NOTRVTVCB, FATAL, 'Not RVT pointer in VCB');
207      IF CH$EQ[ (.NAME[0], .NAME[1],
208      RVT$$_STRUCNAME, RVT[RVT$$_STRUCNAME], ' ' )
209      THEN LEAVE SEARCH_LOOP;
210      END;
211      END;
212      UCB = .UCB[UCB$$_LINK];
213      END;
214      DDB = .DDB[DDB$$_LINK];
215      END;
216      SB = .SB [SB$$_FLINK];
217      END;
218      END;
219      END;
220      END;
221      ! Compute the size RVT needed. If none was found, create one. If the volume
222      ! has a zero RVN, we are creating a new volume set (since we didn't find the
223      ! RVT).
224      !
225      !
226      ENTRY_COUNT = MAXU (.HOME_BLOCK[HM2$$_RVN], .HOME_BLOCK[HM2$$_SETCOUNT]);
227
228      IF .DDB EQL 0
229      THEN
230      BEGIN
231      IF .ENTRY_COUNT EQL 0
232      THEN
233      BEGIN
234      ENTRY_COUNT = 1;
235      HOME_BLOCK[HM2$$_RVN] = 1;
236      HOME_BLOCK[HM2$$_SETCOUNT] = 1;
237      REAL_VCB[VCB$$_RVN] = 1;
238      END;
239
240      RVT = ALLOCATE MEM (MAXU (.ENTRY_COUNT+4, RVT$$_MINSIZE) * 4 + RVT$$_LENGTH, 0);
241      RVT[RVT$$_TYPE] = DYN$C_RVT;
242      RVT[RVT$$_NVOLS] = .ENTRY_COUNT;
243      CH$COPY (.NAME[0], .NAME[1], ' ', RVT$$_STRUCNAME, RVT[RVT$$_STRUCNAME]);
244      END
245
246      ! If we found an RVT, see if it is big enough. If not, allocate a new one,
```



```

248 0778 3 | copy the contents of the old to the new, fix up all pointers, and dispose
249 0779 3 | of the old one. If the volume was not in a set, do the initial bind,
250 0780 3 | assigning the next higher RVN. Note that RVN 1 must be mounted for later
251 0781 3 | use (also assuring that RVT$B_NVOLS contains the size of the set).
252 0782 3 |
253 0783 3 |
254 0784 2 ELSE
255 0785 3 BEGIN
256 0786 3 IF .ENTRY_COUNT EQL 0
257 0787 3 THEN
258 0788 4 BEGIN
259 0789 4 IF .RVT[RVT$L_UCBLST] EQL 0
260 0790 4 THEN
261 0791 5 BEGIN
262 0792 5 UNLOCK IODB ();
263 0793 5 ERR_EXIT (MOUN$_RVN1NOTMT);
264 0794 4 END;
265 0795 4 ENTRY_COUNT = .RVT[RVT$B_NVOLS] + 1;
266 0796 4 IF .ENTRY_COUNT GEQU 256
267 0797 4 THEN
268 0798 5 BEGIN
269 0799 5 UNLOCK IODB ();
270 0800 5 ERR_EXIT (MOUN$_SETLIMIT);
271 0801 4 END;
272 0802 4 HOME_BLOCK[HM2$W_RVN] = .ENTRY_COUNT;
273 0803 4 REAL_VCB[VCB$W_RVN] = .ENTRY_COUNT;
274 0804 3 END;
275 0805 3
276 0806 3 IF .RVT[RVT$W_SIZE] LSSU .ENTRY_COUNT * 4 + RVT$C_LENGTH
277 0807 3 THEN
278 0808 4 BEGIN
279 0809 4
280 0810 4 | ***** The RVT extend/copy code is tied off because the ACP uses RVT
281 0811 4 | addresses in its cache descriptors. This results in the operational
282 0812 4 | restriction of having to dismount and remount the volume set for each
283 0813 4 | fourth volume added after the minimum RVT size (17). Volume sets of
284 0814 4 | this size are at present of low probability.
285 0815 4 |
286 0816 4
287 0817 4 UNLOCK IODB ();
288 0818 4 ERR_EXIT (MOUN$_SETLIMIT);
289 0819 4
290 0820 4 | ***** End of tie-off.
291 0821 4
292 0822 4 NEW_RVT = ALLOCATE MEM (.ENTRY_COUNT * 4 + RVT$C_LENGTH, 0);
293 0823 4 SIZE = .NEW_RVT[RVT$W_SIZE];
294 0824 4 CH$COPY (.RVT[RVT$W_SIZE], .RVT, 0, .SIZE, .NEW_RVT);
295 0825 4 NEW_RVT[RVT$W_SIZE] = .SIZE;
296 0826 4
297 0827 4 INCR J FROM 1 TO .RVT[RVT$B_NVOLS]
298 0828 4 DO
299 0829 5 BEGIN
300 0830 5 UCB = .VECTOR [NEW_RVT[RVT$L_UCBLST], .J-1];
301 0831 5 IF .UCB NEQ 0
302 0832 5 THEN
303 0833 6 BEGIN
304 0834 6 VCB = .UCB[UCB$L_VCB];
```



```

: 305      0835      6      VCB[VCB$L_RVT] = .NEW_RVT;
: 306      0836      6
: 307      0837      6      FCB = .VCB[VCB$L_FCBFL];
: 308      0838      6      UNTIL .FCB EQL VCB[VCB$L_FCBFL]
: 309      0839      6      DO
: 310      0840      7          BEGIN
: 311      0841      7              WINDOW = .FCB[FCB$L_WLFL];
: 312      0842      7              UNTIL .WINDOW EQL FCB[FCB$L_WLFL]
: 313      0843      7              DO
: 314      0844      8                  BEGIN
: 315      0845      8                      WINDOW[VCB$L_RVT] = .NEW_RVT;
: 316      0846      8                      WINDOW = .WINDOW[VCB$L_WLFL];
: 317      0847      7                      END;
: 318      0848      6                  END;
: 319      0849      6                      FCB = .FCB[FCB$L_FCBFL];
: 320      0850      5                  END;
: 321      0851      4          END;
: 322      0852      4      DEALLOCATE_MEM (.RVT, 0);
: 323      0853      4      RVT = .NEW_RVT;
: 324      0854      3      END;
: 325      0855      2      END;
: 326      0856      2
: 327      0857      2      ! Check the UCB pointer we are about to store into. If it is non-zero,
: 328      0858      2      ! a volume with the same RVN is already mounted.
: 329      0859      2      ! Finally fix up the entry count in the RVT and store the new UCB pointer.
: 330      0860      2      !
: 331      0861      2
: 332      0862      2      IF .VECTOR [RVT[RVT$L_UCBLST], .HOME_BLOCK[HM2$W_RVN]-1] NEQ 0
: 333      0863      2      THEN
: 334      0864      3          BEGIN
: 335      0865      3              UNLOCK_IODB ();
: 336      0866      3              ERR_EXIT (MOUN$_DUPRVN);
: 337      0867      2          END;
: 338      0868      2
: 339      0869      2      RVT[RVT$W_REFC] = .RVT[RVT$W_REFC] + 1;
: 340      0870      2      RVT[RVT$B_NVOLS] = MAXU (.RVT[RVT$B_NVOLS], .ENTRY_COUNT);
: 341      0871      2      VECTOR [RVT[RVT$L_UCBLST], .HOME_BLOCK[HM2$W_RVN]-1] = .NEW_UCB;
: 342      0872      2
: 343      0873      2      UNLOCK_IODB ();
: 344      0874      2      CLEANUP_FLAGS[CLF_UNLOCKDB] = 0;
: 345      0875      2      REAL_RVT = .RVT;
: 346      0876      2      RETURN .RVT;
: 347      0877      2
: 348      0878      1      END;

```

! end of routine ENTER_RVT

```

.TITLE  MAKRV
.IDENT  \V04-000\

.EXTRN  MOUNT_OPTIONS, CLEANUP_FLAGS
.EXTRN  REAL_VCB, REAL_RVT
.EXTRN  HOME_BLOCK, SC$GL_CURPCB
.EXTRN  SC$GQ_CONFIG, LOCK_IODB
.EXTRN  UNLOCK_IODB, ALLOCATE_MEM
.EXTRN  DEALLOCATE_MEM, BUG$_NOTDDBDB
.EXTRN  BUG$_NOTUCBUCB, BUG$_NOTVCBUCB
.EXTRN  BUG$_NOTRVTVCB

```

				.PSECT	\$CODE\$,NOWRT,2	
				.ENTRY	ENTER_RVT, Save R2,R3,R4,R5,R6,R7,R8,R9,-	0607
					R10,RT1	
				CALLS	#0, LOCK IODB	0689
				BISB2	#8, CLEANUP_FLAGS+1	0690
				MOVL	@#SCS\$GQ_CONFIG, SB	0692
				CMPL	SB, #SCS\$GQ_CONFIG	0697
				BNEQ	2\$	
				BRW	15\$	
				MOVL	84(SB), DDB	0701
				BNEQ	4\$	0703
				BRW	14\$	
				CMPB	10(DDB), #6	0706
				BEQL	5\$	
				BUGW		0707
				.WORD	<BUG\$ NOTDDBDDB!4>	
				MOVL	4(DDB), UCB	0708
				BEQL	13\$	0710
				CMPB	10(UCB), #16	0712
				BEQL	7\$	
				BUGW		0713
				.WORD	<BUG\$ NOTUCBUCB!4>	
				MOVAB	56(UCB), R0	0715
				BBC	#14, (R0), 12\$	
				BBC	#19, (R0), 12\$	0716
				BLBS	3(R0), 12\$	0717
				BBS	#5, (R0), 12\$	0718
				BBC	#4, MOUNT_OPTIONS, 8\$	0719
				BBC	#23, (R0), 12\$	0720
				MOVL	@#SCH\$GL_CURPCB, R0	0721
				CMPL	44(UCB), -96(R0)	
				BNEQ	12\$	
				BRB	9\$	
				BBS	#23, (R0), 12\$	0722
				MOVL	52(UCB), VCB	0726
				CMPB	10(VCB), #17	0727
				BEQL	10\$	
				BUGW		0728
				.WORD	<BUG\$ NOTVCBUCB!4>	
				MOVL	32(VCB), RVT	0730
				BEQL	12\$	0731
				CMPB	10(RVT), #16	0732
				BEQL	12\$	
				CMPB	10(RVT), #14	0735
				BEQL	11\$	
				BUGW		0736
				.WORD	<BUG\$ NOTRVTVCB!4>	
				MOVL	NAME, -R0	0737
				CMPC5	(R0), @4(R0), #32, #12, 12(RVT)	0738
				BEQL	15\$	
				MOVL	48(UCB), UCB	0742
				BRB	6\$	0710
				MOVL	(DDB), DDB	0744
				BRW	3\$	0703

				OFFC	00000	
0000G	CF	00	FB	00002		
0000G	CF	08	88	00007		
00000000G	55	00000000G	9F	D0	0000C	
	8F		55	D1	00013	1\$:
			03	12	0001A	
			0099	31	0001C	
	54	54	A5	D0	0001F	2\$:
			03	12	00023	3\$:
			008A	31	00025	
	06	0A	A4	91	00028	4\$:
			04	13	0002C	
			FEFF	0002E		
			0000*	00030		
	58	04	A4	D0	00032	5\$:
			74	13	00036	6\$:
	10	0A	A8	91	00038	
			04	13	0003C	
			FEFF	0003E		
			0000*	00040		
	50	38	A8	9E	00042	7\$:
5C	60		0E	E1	00046	
58	60		13	E1	0004A	
	54	03	A0	E8	0004E	
50	60		05	E0	00052	
14	CF		04	E1	00056	
46	60		17	E1	0005C	
	50	00000000G	9F	D0	00060	
	60	A0	A8	D1	00067	
		2C	38	12	0006C	
			04	11	0006E	
	32	60	17	E0	00070	8\$:
		5A	A8	D0	00074	9\$:
		11	AA	91	00078	
			04	13	0007C	
			FEFF	0007E		
			0000*	00080		
	57	20	AA	D0	00082	10\$:
			1E	13	00086	
	10	0A	A7	91	00088	
			18	13	0008C	
	0E	0A	A7	91	0008E	
			04	13	00092	
			FEFF	00094		
			0000*	00096		
	50	04	AC	D0	00098	11\$:
0C	20	04	60	2D	0009C	
		0C	A7		000A2	
			12	13	000A4	
	58	30	A8	D0	000A6	12\$:
			8A	11	000AA	
	54		64	D0	000AC	13\$:
			FF71	31	000AF	

55		65	D0	000B2	14\$:	MOVL	(SB), SB	0747	
		5B	31	000B5		BRW	1\$	0697	
50	0000G	CF	3C	000B8	15\$:	MOVZWL	HOME_BLOCK+38, R0	0757	
50	0000G	CF	B1	000BD		CMPW	HOME_BLOCK+40, R0		
		05	1B	000C2		BLEQU	16\$		
50	0000G	CF	3C	000C4		MOVZWL	HOME_BLOCK+40, R0		
56		50	D0	000C9	16\$:	MOVL	R0, ENTRY_COUNT		
		54	D5	000CC		TSTL	DD8	0759	
		51	12	000CE		BNEQ	19\$		
		56	D5	000D0		TSTL	ENTRY_COUNT	0762	
		15	12	000D2		BNEQ	17\$		
		01	D0	000D4		MOVL	#1, ENTRY_COUNT	0765	
0000G	CF	00010001	8F	D0	000D7	MOVL	#65537, HOME_BLOCK+38	0766	
	50	0000G	CF	D0	000E0	MOVL	REAL_VCB, R0	0768	
OE	A0		01	B0	000E5	MOVW	#1, T4(R0)		
			7E	D4	000E9	17\$:	CLRL	-(SP)	0771
			A6	9E	000EB	MOVAB	4(R6), R0		
			50	D1	000EF	CMP	R0, #18		
			03	1E	000F2	BGEQU	18\$		
			12	D0	000F4	MOVL	#18, R0		
7E			02	78	000F7	18\$:	ASHL	#2, R0, -(SP)	
			8F	C0	000FB	ADDL2	#68, (SP)		
0000G	CF	00000044	02	FB	00102	CALLS	#2, ALLOCATE_MEM		
	57		50	D0	00107	MOVL	R0, RVT		
OA	A7		0E	90	0010A	MOVB	#14, 10(RVT)	0772	
OB	A7		56	90	0010E	MOVB	ENTRY_COUNT, 11(RVT)	0773	
	50		AC	D0	00112	MOVL	NAME, R0	0774	
OC	20	04	B0	2C	00116	MOVCS	(R0), @4(R0), #32, #12, 12(RVT)		
		OC	A7	0011C					
		00D3	31	0011E		BRW	28\$	0759	
			56	D5	00121	19\$:	TSTL	ENTRY_COUNT	0786
			46	12	00123	BNEQ	22\$		
		44	A7	D5	00125	TSTL	68(RVT)	0789	
			12	12	00128	BNEQ	20\$		
	0000G	CF	00	FB	0012A	CALLS	#0, UNLOCK_IODB	0792	
		0072819C	8F	DD	0012F	PUSHL	#7504284	0793	
00000000G	00		01	FB	00135	CALLS	#1, LIB\$STOP		
	56	OB	A7	9A	0013C	20\$:	MOVZBL	11(RVT), ENTRY_COUNT	0795
			56	D6	00140	INCL	ENTRY_COUNT		
00000100	8F		56	D1	00142	CMP	ENTRY_COUNT, #256	0796	
			12	1F	00149	BLSSU	21\$		
	0000G	CF	00	FB	0014B	CALLS	#0, UNLOCK_IODB	0799	
		007281A4	8F	DD	00150	PUSHL	#7504292	0800	
00000000G	00		01	FB	00156	CALLS	#1, LIB\$STOP		
0000G	CF		56	B0	0015D	21\$:	MOVW	ENTRY_COUNT, HOME_BLOCK+38	0802
	50	0000G	CF	D0	00162	MOVL	REAL_VCB, R0	0803	
OE	A0		56	B0	00167	MOVW	ENTRY_COUNT, 14(R0)		
52	56		02	78	0016B	22\$:	ASHL	#2, ENTRY_COUNT, R2	0806
	52	44	A2	9E	0016F	MOVAB	68(R2), R2		
52	10		00	ED	00173	CMPZV	#0, #16, 8(RVT), R2		
			79	1E	00179	BGEQU	28\$		
	0000G	CF	00	FB	0017B	CALLS	#0, UNLOCK_IODB	0817	
		007281A4	8F	DD	00180	PUSHL	#7504292	0818	
00000000G	00		01	FB	00186	CALLS	#1, LIB\$STOP		
			7E	D4	0018D	CLRL	-(SP)	0822	
			52	DD	0018F	PUSHL	R2		
	0000G	CF	02	FB	00191	CALLS	#2, ALLOCATE_MEM		

5B	00	59	08	50	D0	00196	MOVL	R0, NEW_RVT		
		5B	08	A9	3C	00199	MOVZWL	8(NEW_RVT), SIZE	0823	
		67	08	A7	2C	0019D	MOVCS	8(RVT), (RVT), #0, SIZE, (NEW_RVT)	0824	
				69		001A3				
	08	A9		5B	B0	001A4	MOVW	SIZE, 8(NEW_RVT)	0825	
		55	08	A7	9A	001A8	MOVZBL	11(RVT), R5	0827	
		54	44	A9	9E	001AC	MOVAB	68(NEW_RVT), R4	0830	
				50	D4	001B0	CLRL	J		
				30	11	001B2	BRB	27\$		
		58	FC	A4	40	D0	001B4	23\$:		
				29	13	001B9	BEQL	-4(R4)[J], UCB		
		5A	34	A8	D0	001BB	BEQL	27\$	0831	
	20	AA		59	D0	001BF	MOVL	52(UCB), VCB	0834	
		51		6A	D0	001C3	MOVL	NEW_RVT, 32(VCB)	0835	
		5A		51	D1	001C6	MOVL	(VCB), FCB	0837	
				16	13	001C9	CMPL	FCB, VCB	0838	
		52	10	A1	D0	001CB	BEQL	26\$		
		53	10	A1	9E	001CF	MOVL	16(FCB), WINDOW	0841	
		53		52	D1	001D3	MOVAB	16(FCB), R3	0842	
				EE	13	001D6	CMPL	WINDOW, R3		
	1C	A2		59	D0	001D8	BEQL	24\$		
		52		62	D0	001DC	MOVL	NEW_RVT, 28(WINDOW)	0845	
				EE	11	001DF	MOVL	(WINDOW), WINDOW	0846	
		51		61	D0	001E1	BRB	25\$	0842	
	CC	50		55	F3	001E4	MOVL	(FCB), FCB	0849	
				7E	D4	001E8	AOBLEQ	R5, J, 23\$	0827	
				57	DD	001EA	CLRL	-(SP)	0852	
	0000G	CF		02	FB	001EC	PUSHL	RVT		
		57		59	D0	001F1	CALLS	#2, DEALLOCATE_MEM		
		50	0000G	CF	3C	001F4	MOVL	NEW_RVT, RVT	0853	
			40	A7	40	D5	MOVZWL	HOME_BLOCK+38, R0	0862	
				12	13	001FD	TSTL	64(RVT)[R0]		
				00	FB	001FF	BEQL	29\$		
	0000G	CF		8F	DD	00204	CALLS	#0, UNLOCK_IODB	0865	
			007281BC	01	FB	0020A	PUSHL	#7504316	0866	
	00000000G	00		04	A7	B6	CALLS	#1, LIB\$STOP		
		50	04	A7	9A	00211	INCW	4(RVT)	0869	
		56	0B	A7	9A	00214	MOVZBL	11(RVT), R0	0870	
				50	D1	00218	CMPL	R0, ENTRY_COUNT		
				03	1E	0021B	BGEQU	30\$		
		50		56	D0	0021D	MOVL	ENTRY_COUNT, R0		
	0B	A7		50	90	00220	MOVB	R0, 1T(RVT)		
		50	0000G	CF	3C	00224	MOVZWL	HOME_BLOCK+38, R0	0871	
	40	A7	08	AC	D0	00229	MOVL	NEW_UCB, 64(RVT)[R0]		
				00	FB	0022F	CALLS	#0, UNLOCK_IODB	0873	
	0000G	CF		08	8A	00234	BICB2	#8, CLEANUP_FLAGS+1	0874	
	0000G	CF		57	D0	00239	MOVL	RVT, REAL_RVT	0875	
		50		57	D0	0023E	MOVL	RVT, R0	0876	
				04	00241		RET		0878	

; Routine Size: 578 bytes, Routine Base: \$CODE\$ + 0000

:	349	0879	1	
:	350	0880	1	END
:	351	0881	0	ELUDOM

.EXTRN LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	578	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	50	0	1000	00:01.9

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MAKRV/OBJ=OBJ\$:MAKRV MSRC\$:MAKRV/UPDATE=(ENH\$:MAKRV)

: Size: 578 code + 0 data bytes
: Run Time: 00:19.0
: Elapsed Time: 00:36.8
: Lines/CPU Min: 2783
: Lexemes/CPU-Min: 29143
: Memory Used: 224 pages
: Compilation Complete

0244

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY