

[illegible]

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```



```
0001 0 MODULE MWTUR2 (  
0002 0     LANGUAGE (BLISS32),  
0003 0     IDENT = 'V04-000'  
0004 0 ) =  
0005 1 BEGIN  
0006 1  
0007 1  
0008 1 *****  
0009 1 *  
0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0012 1 * ALL RIGHTS RESERVED.  
0013 1 *  
0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0019 1 * TRANSFERRED.  
0020 1 *  
0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0023 1 * CORPORATION.  
0024 1 *  
0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0027 1 *  
0028 1 *****  
0029 1  
0030 1 ++  
0031 1  
0032 1  
0033 1 FACILITY: MOUNT Utility Structure Level 2  
0034 1  
0035 1 ABSTRACT:  
0036 1  
0037 1     This module generates a window mapping the desired VBN from  
0038 1     the supplied file header. This module is a direct crib from the  
0039 1     FCP module WITURN. When we get conditional compilation with different  
0040 1     parameter files worked out, they should really be the same source.  
0041 1  
0042 1 ENVIRONMENT:  
0043 1  
0044 1     STARLET operating system, including privileged system services  
0045 1     and internal exec routines.  
0046 1  
0047 1 --  
0048 1  
0049 1  
0050 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 7-Dec-1976 14:38  
0051 1  
0052 1 MODIFIED BY:  
0053 1  
0054 1     V03-001 HH0041 Hai Huang 24-Jul-1984  
0055 1     Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.  
0056 1  
0057 1     V02-001 ACG0167 Andrew C. Goldstein, 18-Apr-1980 13:39
```

MWTUR2
V04-000

1 3
16-Sep-1984 01:27:16
14-Sep-1984 12:45:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]MWTUR2.B32;1 Page 2 (1)

```
: 58
: 59
: 60
: 61
: 62
: 63

0058 1 !
0059 1 !**
0060 1
0061 1
0062 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
0063 1 REQUIRE 'SRC$:MOUDEF.B32';
```

Previous revision history moved to MOUNT.REV


```

65 0595 1 GLOBAL ROUTINE TURN_WINDOW2 (WINDOW, HEADER, DESIRED_VBN, START_VBN, RVN) =
66 0596 1
67 0597 1 ++
68 0598 1
69 0599 1 FUNCTIONAL DESCRIPTION:
70 0600 1
71 0601 1 This routine scans the map area of the supplied file header
72 0602 1 and builds retrieval pointers in the window until
73 0603 1 (1) the entire header has been scanned, or
74 0604 1 (2) the first retrieval pointer in the window maps the desired VBN
75 0605 1
76 0606 1 CALLING SEQUENCE:
77 0607 1 TURN_WINDOW (ARG1, ARG2, ARG3, ARG4, ARG5)
78 0608 1
79 0609 1 INPUT PARAMETERS:
80 0610 1 ARG1: address of window block
81 0611 1 ARG2: address of file header
82 0612 1 ARG3: desired VBN
83 0613 1 ARG4: starting VBN of file header
84 0614 1 ARG5: RVN of file header
85 0615 1
86 0616 1 IMPLICIT INPUTS:
87 0617 1 NONE
88 0618 1
89 0619 1 OUTPUT PARAMETERS:
90 0620 1 updated window
91 0621 1
92 0622 1 IMPLICIT OUTPUTS:
93 0623 1 NONE
94 0624 1
95 0625 1 ROUTINE VALUE:
96 0626 1 1
97 0627 1
98 0628 1 SIDE EFFECTS:
99 0629 1 NONE
100 0630 1
101 0631 1 --
102 0632 1
103 0633 2 BEGIN
104 0634 2
105 0635 2 MAP
106 0636 2 WINDOW : REF BBLOCK, ! pointer to window
107 0637 2 HEADER : REF BBLOCK; ! pointer to file header
108 0638 2
109 0639 2 LINKAGE
110 0640 2 L_MAP_POINTER = JSB :
111 0641 2 GLOBAL (COUNT = 6, LBN = 7, MAP_POINTER = 8);
112 0642 2
113 0643 2 GLOBAL REGISTER
114 0644 2 COUNT = 6, ! retrieval pointer count
115 0645 2 LBN = 7, ! retrieval pointer start LBN
116 0646 2 MAP_POINTER = 8 : REF BBLOCK; ! pointer to scan header map area
117 0647 2
118 0648 2 LABEL
119 0649 2 MAP_BUILD; ! loop to build window map
120 0650 2
121 0651 2 LOCAL
```

```
122 0652 2 VBN, ! VBN in scanning window
123 0653 2 COUNTER, ! loop counter
124 0654 2 W_POINTER : REF BBLOCK; ! pointer to scan window
125 0655 2
126 0656 2 MACRO
127 0657 2 WINDOW_MAP = (.WINDOW+WCB$C_MAP)%; !start of window map area
128 0658 2
129 0659 2 EXTERNAL ROUTINE
130 0660 2 GET_MAP_POINTER : L_MAP_POINTER; ! get value of next header map pointer
131 0661 2
132 0662 2 ! Scan the window looking for the starting VBN of the header. If it is
133 0663 2 ! contained within the window, truncate the window so that it maps exactly
134 0664 2 ! up to the start of the header. If the starting VBN is not contained in the
135 0665 2 ! window, the entire window must be discarded. However, if the desired VBN
136 0666 2 ! precedes the header start VBN, we do nothing since the window is already
137 0667 2 ! best effort.
138 0668 2
139 0669 2
140 0670 2 W_POINTER = WINDOW_MAP; ! point to first retrieval pointer
141 0671 2 VBN = .WINDOW[WCB$C_STVBN]; ! get starting VBN of window
142 0672 2
143 0673 2 IF .START_VBN LEQU .VBN
144 0674 2 OR
145 0675 2 BEGIN
146 0676 2 INCR J FROM 1 TO .WINDOW[WCB$W_NMAP]
147 0677 2 DO
148 0678 2 BEGIN
149 0679 2 VBN = .VBN + .W_POINTER[WCB$W_COUNT]; ! VBN at end of this pointer
150 0680 2 W_POINTER = .W_POINTER + 6;
151 0681 2 IF .START_VBN EQL .VBN
152 0682 2 THEN
153 0683 2 BEGIN
154 0684 2 WINDOW[WCB$W_NMAP] = .J; ! truncate the window
155 0685 2 EXITLOOP 0;
156 0686 2 END;
157 0687 2 END
158 0688 2 END
159 0689 2
160 0690 2 THEN ! header VBN is not in window
161 0691 2 BEGIN
162 0692 2 IF .DESIRED_VBN LSSU .START_VBN
163 0693 2 AND .START_VBN GTRU 1
164 0694 2 THEN
165 0695 2 RETURN 1 ! leave it alone
166 0696 2 ELSE
167 0697 2 BEGIN
168 0698 2 WINDOW[WCB$W_NMAP] = 0; ! flush the window
169 0699 2 WINDOW[WCB$L_STVBN] = .START_VBN;
170 0700 2 W_POINTER = WINDOW_MAP; ! point to first pointer
171 0701 2 END;
172 0702 2 END;
173 0703 2
174 0704 2 ! The window is now suitably initialized. Set up necessary pointers.
175 0705 2 ! Now scan the map area, extracting retrieval pointers.
176 0706 2
177 0707 2
178 0708 2 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2; ! point to map area
```



```
179 0709 2
180 0710 3 MAP_BUILD: BEGIN
181 0711 3 UNTIL .MAP_POINTER GEQA .HEADER + (.HEADER[FH2$B_MPOFFSET] + .HEADER[FH2$B_MAP_INUSE]) * 2
182 0712 3 DO
183 0713 4 BEGIN
184 0714 4
185 0715 4 GET_MAP_POINTER ();
186 0716 4
187 0717 4 ! Build new retrieval pointers, using as many as needed to run out the
188 0718 4 count. If the window is full, shuffle the entries up by one. If this
189 0719 4 would cause the pointer mapping the desired VBN to fall off the top,
190 0720 4 we are done.
191 0721 4 !
192 0722 4
193 0723 4 IF .COUNT NEQ 0
194 0724 4 THEN
195 0725 4 WHILE 1 DO
196 0726 5 BEGIN
197 0727 5 IF (.WINDOW[WCB$W_NMAP]+1)*6 + WCB$C_LENGTH
198 0728 5 GTRU .WINDOW[WCB$W_SIZE]
199 0729 5 THEN
200 0730 6 BEGIN
201 0731 6 IF .WINDOW[WCB$W_STVBN] + .WINDOW[WCB$W_P1_COUNT] GTRU .DESIRED_VBN
202 0732 6 THEN LEAVE MAP_BUILD;
203 0733 6
204 0734 6 WINDOW[WCB$W_NMAP] = .WINDOW[WCB$W_NMAP] - 1;
205 0735 6 WINDOW[WCB$W_STVBN] = .WINDOW[WCB$W_STVBN] + .WINDOW[WCB$W_P1_COUNT];
206 0736 6 CHSMOVE (.WINDOW[WCB$W_NMAP]*6, WINDOW_MAP+6, WINDOW_MAP);
207 0737 6 W_POINTER = .W_POINTER - 6;
208 0738 5 END;
209 0739 5
210 0740 5 ! Finally build the pointer and count it.
211 0741 5 !
212 0742 5
213 0743 5 W_POINTER[WCB$W_COUNT] = MINU (.COUNT, 65535);
214 0744 5 W_POINTER[WCB$W_LBN] = .LBN;
215 0745 5 (W_POINTER[WCB$W_LBN]<24,8> = .RVN;
216 0746 5 W_POINTER = .W_POINTER + 6;
217 0747 5 WINDOW[WCB$W_NMAP] = .WINDOW[WCB$W_NMAP] + 1;
218 0748 5 LBN = .LBN + 65535;
219 0749 5 COUNT = .COUNT - MINU (.COUNT, 65535);
220 0750 5 IF .COUNT EQL 0 THEN EXITLOOP;
221 0751 4 END;
222 0752 4
223 0753 3 END;
224 0754 2 END;
225 0755 2
226 0756 2 RETURN 1;
227 0757 2
228 0758 1 END;
```

```
! end of header scan loop
! end of block MAP_BUILD
```

```
! end of routine TURN_WINDOW2
```

```
.TITLE MWTUR2
.IDENT \V04-000\
.EXTRN GET_MAP_POINTER
```

				.PSECT	\$CODE\$,NOWRT,2	
			OFFC 00000	.ENTRY	TURN WINDOW2, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	0595
5E		04	C2 00002	SUBL2	R10,R11	
59		04	AC D0 00005	MOVL	#4, SP	
		30	A9 9F 00009	PUSHAB	WINDOW, R9	0670
			6E DD 0000C	PUSHL	48(R9)	
52		2C	A9 D0 0000E	PUSHL	(SP)	
50		10	AC D0 00012	MOVL	44(R9), VBN	0671
52			50 D1 00016	MOVL	START VBN, R0	0673
			21 1B 00019	CMPL	R0, VBN	
53		16	A9 3C 0001B	BLEQU	3\$	
			51 D4 0001F	MOVZWL	22(R9), R3	0676
			15 11 00021	CLRL	J	
54		00	BE 3C 00023 1\$:	BRB	2\$	
52			54 C0 00027	MOVZWL	@W_POINTER, R4	0679
6E			06 C0 0002A	ADDL2	R4, VBN	
52			50 D1 0002D	ADDL2	#6, W_POINTER	0680
			06 12 00030	CMPL	R0, VBN	0681
			51 B0 00032	BNEQ	2\$	
16	A9		1A 11 00036	MOVW	J, 22(R9)	0684
			53 F3 00038 2\$:	BRB	5\$	0685
51			AC D1 0003C 3\$:	AOBLEQ	R3, J, 1\$	0676
50		0C	05 1E 00040	CMPL	DESIRED_VBN, R0	0692
			50 D1 00042	BGEQU	4\$	
01			55 1A 00045	CMPL	R0, #1	0693
		16	A9 B4 00047 4\$:	BGTRU	8\$	
			50 D0 0004A	CLRW	22(R9)	0698
2C	A9		04 AE D0 0004E	MOVL	R0, 44(R9)	0699
		04	AC D0 00052 5\$:	MOVL	4(SP), W_POINTER	0700
08	5B	08	AB 9A 00056	MOVL	HEADER, R11	0708
	AE	01	AE D0 0005B	MOVZBL	1(R11), 8(SP)	
50		08	6B40 3E 0005F	MOVL	8(SP), R0	
58			AB 9A 00063 6\$:	MOVAW	(R11)[R0], MAP_POINTER	
50		3A	AE C0 00067	MOVZBL	58(R11), R0	0711
50		08	6B40 3E 0006B	ADDL2	8(SP), R0	
50			58 D1 0006F	MOVAW	(R11)[R0], R0	
			7B 1E 00072	CMPL	MAP_POINTER, R0	
		0000G	30 00074	BGEQU	11\$	
			56 D5 00077	BSBW	GET MAP_POINTER	0715
			E8 13 00079	TSTL	COUNT	0723
5A		16	A9 9E 0007B	BEQL	6\$	
50			6A 3C 0007F 7\$:	MOVAB	22(R9), R10	0727
50			06 C4 00082	MOVZWL	(R10), R0	
50			36 C0 00085	MULL2	#6, R0	
10			00 ED 00088	ADDL2	#54, R0	
			27 1E 0008E	CMPZV	#0, #16, 8(R9), R0	0728
			BE 3C 00090	BGEQU	9\$	
50		04	A9 C0 00094	MOVZWL	@4(SP), R0	0731
50		2C	50 D1 00098	ADDL2	44(R9), R0	
0C	AC		51 1A 0009C 8\$:	CMPL	R0, DESIRED_VBN	
			6A B7 0009E	BGTRU	11\$	
			BE 3C 000A0	DECW	(R10)	0734
50		04	50 C0 000A4	MOVZWL	@4(SP), R0	0735
50			6A 3C 000AB	ADDL2	R0, 44(R9)	
50			06 C4 000AB	MOVZWL	(R10), R0	0736
				MULL2	#6, R0	

04	BE	36	A9	50	28	000AE	MOVC3	R0, 54(R9), @4(SP)	
			6E	06	C2	000B4	SUBL2	#6, W_POINTER	0737
			50	56	D0	000B7	MOVL	COUNT, R0	0743
	0000FFFF		8F	50	D1	000BA	CMPL	R0, #65535	
				05	1B	000C1	BLEQU	10\$	
			50	8F	3C	000C3	MOVZWL	#65535, R0	
		00	BE	50	B0	000C8	MOVW	R0, @W_POINTER	
51			6E	02	C1	000CC	ADDL3	#2, W_POINTER, R1	0744
			61	87	7E	000D0	MOVAQ	(LBN)7, (R1)	
51			6E	05	C1	000D3	ADDL3	#5, W_POINTER, R1	0745
			61	AC	90	000D7	MOVB	RVN, (R1)	
			6E	06	C0	000DB	ADDL2	#6, W_POINTER	0746
				6A	B6	000DE	INCW	(R10)	0747
			57	E7	9E	000E0	MOVAB	65527(R7), LBN	0748
			56	50	C2	000E7	SUBL2	R0, COUNT	0749
				93	12	000EA	BNEQ	7\$	0750
				FF74	31	000EC	BRW	6\$	
			50	01	D0	000EF	MOVL	#1, R0	0756
				04	000F2	RET			0758

; Routine Size: 243 bytes, Routine Base: \$CODE\$ + 0000

: 229 0759 1
: 230 0760 1 END
: 231 0761 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	243	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	18	0	1000	00:02.0

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MWTUR2/OBJ=OBJ\$:MWTUR2 MSRC\$:MWTUR2/UPDATE=(ENH\$:MWTUR2)

MWTUR2
V04-000

B 4
16-Sep-1984 01:27:16
14-Sep-1984 12:45:33

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]MWTUR2.B32;1 Page 8 (2)

; Size: 243 code + 0 data bytes
; Run Time: 00:14.4
; Elapsed Time: 00:30.0
; Lines/CPU Min: 3179
; Lexemes/CPU-Min: 29594
; Memory Used: 144 pages
; Compilation Complete

REB
V04

0246 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

