


```

FFFFFFFFF 000000 RRRRRRRR CCCCCCCC BBBB8888
FFFFFFFFF 000000 RRRRRRRR CCCCCCCC BBBB8888
FF        00      00 RR      RR CC      BB      BB
FF        00      00 RR      RR CC      BB      BB
FF        00      00 RR      RR CC      BB      BB
FF        00      00 RR      RR CC      BB      BB
FFFFFFFFF 00      00 RRRRRRRR CC      BBBB8888
FFFFFFFFF 00      00 RRRRRRRR CC      BBBB8888
FF        00      00 RR  RR  CC      BB      BB
FF        00      00 RR  RR  CC      BB      BB
FF        00      00 RR  RR  CC      BB      BB
FF        00      00 RR  RR  CC      BB      BB
FF        000000 RR      RR CCCCCCCC BBBB8888
FF        000000 RR      RR CCCCCCCC BBBB8888

```

```

LL        111111 SSSSSSSS
LL        111111 SSSSSSSS
LL        11      SS
LL        11      SS
LL        11      SS
LL        11      SS
LL        11      SSSSSS
LL        11      SSSSSS
LL        11      SS
LL        11      SS
LL        11      SS
LL        11      SS
LLLLLLLLL 111111 SSSSSSSS
LLLLLLLLL 111111 SSSSSSSS

```



```

1 0001 0 MODULE FOR$$CB (%TITLE 'Push, Pop, Allocate, and deallocate LUB/ISB/RAB'
2 0002 0 IDENT = '2-005' ! File: FORCB.B32 Edit: LEB2005
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 * ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 * TRANSFERRED. *
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 * CORPORATION. *
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1 ++
30 0030 1 FACILITY: language support library
31 0031 1
32 0032 1 ABSTRACT:
33 0033 1
34 0034 1 This module interfaces to FOR$$CCB DATA to allocate,
35 0035 1 deallocate, push and pop the LUB/ISB/RAB data structure, which
36 0036 1 is central to the I/O system.
37 0037 1
38 0038 1 ENVIRONMENT: User mode, AST level or not or mixed
39 0039 1
40 0040 1 AUTHOR: Thomas N. Hastings, CREATION DATE: 01-June-77
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 Thomas N. Hastings, 01-June-77: VERSION 01
45 0045 1 [Previous edit history removed. SBL 24-Sept-1982]
46 0046 1 1-032 - Remove AST reentrancy window by performing IOINPROG interlock before
47 0047 1 LUN_OWNR test in FOR$$CB_PUSH. Replace individual zeroing of ISB
48 0048 1 bits with a zero of the word in which they are contained for better
49 0049 1 code. Use a new structure for OTSS$V_LUN_OWNR for smaller code.
50 0050 1 SBL 25-Sept-1980
51 0051 1 1-033 - Include secondary message FOR$ IO_NONFOR when signaling
52 0052 1 FOR$K_MIXFILACC. JAW 29-Aug-1981
53 0053 1 1-034 - Clear OTSS$V_IOINPROG before signaling FOR$K_MIXFILACC, to
54 0054 1 ensure that unit is left in a consistent state. SPR 11-38566.
55 0055 1 JAW 29-Aug-1981
56 0056 1 1-035 - Replace $DESCRIPTOR in edit 1-033 with UPLIT to keep code PIC.
57 0057 1 JAW 31-Aug-1981

```

```
58 0058 1 : 1-036 - Add missing external declarations. SBL 2-Dec-1981
59 0059 1 : 2-001 - Remove all references to OTSS$ routines and data structures.
60 0060 1 : The data structures are now FORTRAN-only, although their layout
61 0061 1 : and use is still in parallel with BASIC's. Change to use
62 0062 1 : prologue file, and general cleanup for inclusion in FORRTL
63 0063 1 : shared image. SBL 24-Sept-1982
64 0064 1 : 2-002 - Use ISB$A PREVIOUS LUB for backlink to previous LUB instead of
65 0065 1 : second entry in FOR$SAA LUB_TAB. Add logic to allow simultaneous
66 0066 1 : ENCODE/DECODE/Internal File operations. SBL 2-Dec-1982
67 0067 1 : 2-003 - Allocate FAB and NAM along with RAB and rest of CCB from heap.
68 0068 1 : SBL 17-Jan-1983
69 0069 1 : 2-004 - Deallocate RFA cache if necessary. SBL 2-June-1983
70 0070 1 : 2-005 - Change reference in DEALLOCATE from LUB$A RFA CACHE_PTR to
71 0071 1 : LUB$A RFA CACHE_BEG to fix problem with BACKSPACE.
72 0072 1 : LEB 27-Jan-1984
73 0073 1 : --
74 0074 1 :
```

```
76 0075 1 %SBTTL'Declarations'
77 0076 1
78 0077 1 : PROLOGUE FILE:
79 0078 1
80 0079 1
81 0080 1 REQUIRE 'RTLIN:FORPROLOG';           ! Structure and symbol definitions
82 0146 1
83 0147 1
84 0148 1 : TABLE OF CONTENTS:
85 0149 1
86 0150 1
87 0151 1 FORWARD ROUTINE
88 0152 1   FOR$$CB_PUSH : JSB_CB_PUSH NOVALUE,       ! Allocate or find LUB/ISB/RAB - beg of each I/O statment
89 0153 1   ALLOCATE : CALL_CCB NOVALUE,           ! Allocate CCB
90 0154 1   FOR$$CB_POP : JSB_CB_POP NOVALUE,     ! Pop LUB/ISB/RAB - end of each I/O statement
91 0155 1   DEALLOCATE : CALL_CCB NOVALUE,        ! Deallocate CCB
92 0156 1   FOR$$CB_GET : JSB_CB_GET NOVALUE,     ! Get current LUB/ISB/RAB (called by non-shared code only)
93 0157 1   FOR$$CB_FETCH : CALL_CCB NOVALUE,    ! Fetch a LUB, or 0
94 0158 1   FOR$$NEXT_LUN : NOVALUE,             ! Get next FORTRAN LUN.
95 0159 1   FOR$$FP_MATCH : CALL_CCB NOVALUE,    ! Get CCB that matches FP
96 0160 1   INITIALIZE_INTFIL_QUEUE: NOVALUE;    ! Initialize INTFIL_QUEUE
97 0161 1
98 0162 1
99 0163 1 :+ Include FOR$$CB_RET as a synonym for FOR$$CB_POP to maintain
100 0164 1 : compatability with old versions of FOR$$ERROR.
101 0165 1 :-
102 0166 1
103 0167 1 GLOBAL BIND
104 0168 1   ROUTINE
105 0169 1   FOR$$CB_RET = FOR$$CB_POP : JSB_CB_POP NOVALUE;
106 0170 1
107 0171 1
108 0172 1 : GLOBAL STORAGE:
109 0173 1
110 0174 1
111 0175 1 GLOBAL
112 0176 1   FOR$$A_CUR_LUB : INITIAL (0);           ! Contains the address of the current LUB
113 0177 1
114 0178 1
115 0179 1 :+ The following structure is used for addressing FOR$$AA_LUB_TAB.
116 0180 1 : It is similar to VECTOR, but offsets the index so that
117 0181 1 : negative logical unit numbers can be used.
118 0182 1 :-
119 0183 1
120 0184 1 STRUCTURE
121 0185 1   FOR$$LUB_TAB_ST [I; N, LB, UNIT = 4, EXT = 0] =
122 0186 1   [N*UNIT]
123 0187 1   (FOR$$LUB_TAB_ST + ((I - LB)*UNIT))<0, %BPUNIT*UNIT, EXT>;
124 0188 1
125 0189 1 :+
126 0190 1 : The following table of longwords is used to associate LUB addresses with
127 0191 1 : unit numbers. Each entry contains 0 if there is no
128 0192 1 : LUB, or the address of the LUB.
129 0193 1 :-
130 0194 1
131 0195 1 GLOBAL
132 0196 1   FOR$$AA_LUB_TAB : VOLATILE FOR$$LUB_TAB_ST
```

```

133 0197 1      [-LUB$K_ILUN_MIN + LUB$K_LUN_MAX + 1, LUB$K_ILUN_MIN];
134 0198 1
135 0199 1
136 0200 1      OWN STORAGE:
137 0201 1
138 0202 1
139 0203 1
140 0204 1      +
141 0205 1      Each bit of the following BITVECTOR corresponds to a LUN. The bit is
142 0206 1      set if there is any I/O activity outstanding for the LUN. The bit
143 0207 1      must be kept here rather than in the LUB because there can be I/O
144 0208 1      activity outstanding even before the LUB is allocated.
145 0209 1
146 0210 1      The name FOR$$V_IOINPROG is bound to the appropriate offset in the
147 0211 1      bitvector so that the correct bit can be directly addressed by unit number.
148 0212 1
149 0213 1      OWN
150 0214 1      IOINPROG_VECTOR : VOLATILE BITVECTOR
151 0215 1      [((-LUB$K_ILUN_MIN + LUB$K_LUN_MAX + %BPVAL)/%BPVAL)*%BPVAL];
152 0216 1      BIND
153 0217 1      FOR$$V_IOINPROG = IOINPROG_VECTOR [((7-LUB$K_ILUN_MIN)/8)*8]:
154 0218 1      VOLATILE BITVECTOR [];
155 0219 1
156 0220 1      +
157 0221 1      The following is a queue (non-interlocked) that holds LUBs for ENCODE/DECODE
158 0222 1      and internal file operations. This permits more than one of these operations
159 0223 1      to be active simultaneously.
160 0224 1
161 0225 1
162 0226 1      OWN
163 0227 1      INTFIL_QUEUE: VOLATILE VECTOR [2] INITIAL (0,0),
164 0228 1      V_INTFIL_QUEUE_INIT: VOLATILE INITIAL (0); ! 1 when queue initialized
165 0229 1
166 0230 1
167 0231 1      EXTERNAL REFERENCES:
168 0232 1
169 0233 1
170 0234 1      EXTERNAL ROUTINE
171 0235 1      FOR$$ERRSNS_SAV : NOVALUE,
172 0236 1      FOR$$SIG_NO_LUB : NOVALUE,
173 0237 1
174 0238 1
175 0239 1      FOR$$SIG_DATCOR : NOVALUE,
176 0240 1
177 0241 1
178 0242 1      FOR$$SIGNAL_STO : NOVALUE,
179 0243 1      FOR$$GET_VM,
180 0244 1      FOR$$FREE_VM : NOVALUE;
181 0245 1

```

```

! convert FORTRAN err # to 32-bit code
! Pass LUN explicitly since no current LUB.
! and call LIB$STOP. should never return
! SIGNAL_STOP OT$$ INTDATCOR (INTERNAL
! DATA CORRUPTED IN RUN-TIME LIBRARY)
! in FORTRAN environment
! Signal a fatal FORTRAN error
! Get virtual memory
! Free virtual memory

```

```

183 0246 1 GLOBAL ROUTINE FOR$$CB_PUSH (%SBTTL'Allocate or find CCB'
184 0247 1     LOGICAL UNIT,           ! Logical unit no. (by-value)
185 0248 1     LUN_MIN)           ! Minimum logical unit number (by-value)
186 0249 1     : JSB_CB_PUSH NOVALUE =
187 0250 1
188 0251 1
189 0252 1     **
190 0253 1     FUNCTIONAL DESCRIPTION:
191 0254 1     FOR$$CB_PUSH checks for legal logical UNIT number
192 0255 1     which varies depending on whether this is OPEN or
193 0256 1     default open.  If logical_unit already has
194 0257 1     a LUB/ISB/RAB allocated, only part of the per I/O statement part
195 0258 1     of LUB/ISB/RAB is cleared, namely just the status bits in ISB.
196 0259 1     Otherwise virtual memory is allocated for this logical unit
197 0260 1     and the entire block is initialized to 0.  Then the allocated address
198 0261 1     is remembered in OWN table FOR$$A_LUB_TAB indexed by
199 0262 1     logical_unit.  The RAB is initialized to constants which
200 0263 1     do not change during execution.
201 0264 1
202 0265 1     If an I/O statement on this unit is already in progress, this
203 0266 1     routine signals an error and does not return.
204 0267 1
205 0268 1     CALLING SEQUENCE:
206 0269 1
207 0270 1     JSB FOR$$CB_PUSH (R2=logical_unit.rl.v, R0=lun_min.rl.v)
208 0271 1
209 0272 1     FORMAL PARAMETERS:
210 0273 1
211 0274 1     LOGICAL_UNIT.rl.v     Value of logical unit for which LUB/ISB/RAB is desired (signed)
212 0275 1                     May be negative for TYPE, ACCEPT, READ, PRINT
213 0276 1     LUN_MIN.rl.v         Value of minimum legal logical unit number (signed)
214 0277 1                     Since in a register, must be present.
215 0278 1
216 0279 1     IMPLICIT INPUTS:
217 0280 1
218 0281 1     FOR$$AA_LUB_TAB[logical_unit]  Adr. of LUB/ISB/RAB or 0 for
219 0282 1     this unit
220 0283 1     FOR$$V_IOINPROG[logical unit]  I/O in progress flag
221 0284 1
222 0285 1     IMPLICIT OUTPUTS:
223 0286 1
224 0287 1     CCB                     Base pointer set to adr. of LUB/ISB/RAB for logical_unit.
225 0288 1     FOR$$AA_LUB_TAB[logical_unit]  Adr. of LUB/ISB/RAB for logical_unit
226 0289 1     LUB$W_LUN                signed logical unit number
227 0290 1     RAB$B_BID
228 0291 1     RAB$B_BLN
229 0292 1     RAB$V_TPT                1
230 0293 1     RAB$V_RAH                1
231 0294 1     RAB$V_WBH                1
232 0295 1     RAB$V_LOC                1
233 0296 1
234 0297 1     ROUTINE VALUE:
235 0298 1
236 0299 1     None
237 0300 1
238 0301 1     SIDE EFFECTS:
239 0302 1

```

```
240 0303 1 | Allocates virtual memory if needed.
241 0304 1 | SIGNAL_STOPs FOR$_RECIO_OPE (40='RECURSIVE I/O OPERATION') if
242 0305 1 | logical unit already is in the middle of an I/O statement
243 0306 1 | SIGNAL_STOPs FOR$_INVLOGUNI (32='INVALID LOGICAL UNIT NUMBER')
244 0307 1 | if logical unit is out of range.
245 0308 1 | SIGNAL_STOPs FOR$_INSVIRMEM (41='INSUFFICIENT VIRTUAL MEMORY')
246 0309 1 | if cannot expand program region if needed.
247 0310 1 | --
248 0311 1 |
249 0312 2 | BEGIN
250 0313 2 |
251 0314 2 | BUILTIN
252 0315 2 | TESTBITSS;
253 0316 2 |
254 0317 2 | EXTERNAL REGISTER
255 0318 2 | CCB : REF $FOR$CCB_DECL;
256 0319 2 |
257 0320 2 | !+
258 0321 2 | ! Check range of logical unit. If out of range,
259 0322 2 | ! SIGNAL_STOP FOR$_INVLOGUNI (32='INVALID LOGICAL UNIT NUMBER')
260 0323 2 | !-
261 0324 2 |
262 0325 3 | IF ((.LOGICAL_UNIT GTR LUB$K_LUN_MAX) OR (.LOGICAL_UNIT LSS .LUN_MIN))
263 0326 2 | THEN
264 0327 3 | BEGIN
265 0328 3 | FOR$$SIG_NO_LUB (FOR$K_INVLOGUNI, .LOGICAL_UNIT);
266 0329 3 | RETURN;
267 0330 2 | END;
268 0331 2 |
269 0332 2 | !+
270 0333 2 | ! Test and set IO in progress interlock before doing anything else!
271 0334 2 | ! If this is ENCODE/DECODE/Internal File, ignore interlock.
272 0335 2 | !-
273 0336 2 |
274 0337 3 | IF (TESTBITSS (FOR$$V_IOINPROG [.LOGICAL_UNIT]))
275 0338 2 | THEN
276 0339 2 | IF .LOGICAL_UNIT NEQ LUB$K_LUN_ENCD
277 0340 2 | THEN
278 0341 3 | BEGIN
279 0342 3 | FOR$$SIG_NO_LUB (FOR$K_RECIO_OPE, .LOGICAL_UNIT);
280 0343 3 | RETURN;
281 0344 2 | END;
282 0345 2 |
283 0346 2 | !+
284 0347 2 | ! The following assignment generates no code, but it causes BLISS to generate
285 0348 2 | ! optimal code for the remainder of the routine by preventing the CSE
286 0349 2 | ! .LOGICAL_UNIT-LUB$K_ILUN_MIN from being bound to R2. Thanks, and a tip
287 0350 2 | ! of the keyboard to Steve Hobbs.
288 0351 2 | !-
289 0352 2 |
290 0353 2 | LOGICAL_UNIT = .LOGICAL_UNIT;
291 0354 2 |
292 0355 2 | !+
293 0356 2 | ! Get the CCB address for this unit.
294 0357 2 | !-
295 0358 2 |
296 0359 2 | CCB = .FOR$$AA_LUB_TAB [.LOGICAL_UNIT];
```



```

297 0360 2
298 0361 2
299 0362 2
300 0363 2
301 0364 2
302 0365 2
303 0366 2
304 0367 2
305 0368 2
306 0369 2
307 0370 2
308 0371 2
309 0372 2
310 0373 2
311 0374 2
312 0375 2
313 0376 2
314 0377 2
315 0378 2
316 0379 2
317 0380 2
318 0381 2
319 0382 2
320 0383 2
321 0384 2
322 0385 2
323 0386 2
324 0387 2
325 0388 2
326 0389 2
327 0390 2
328 0391 2
329 0392 2
330 0393 2
331 0394 2
332 0395 2
333 0396 2
334 0397 2
335 0398 2
336 0399 2
337 0400 1

```

```

!+
!- Allocate a LUB/ISB/RAB if necessary.
!-
IF .CCB EQLA 0
THEN
  ALLOCATE (.LOGICAL_UNIT)
ELSE
!+
!- LUB/ISB/RAB already allocated. Perform sanity check.
  BEGIN
    IF ((.CCB [LUB$W_LUN] NEQU .LOGICAL_UNIT<0,16,1>) OR
        (.CCB [RAB$B_BID] NEQU RAB$C_BID))
    THEN
      FOR$$SIG_DATCOR ();
    END;
!+
!- Initialize certain ISB fields, to save FOR$$IO_BEG the trouble.
!-
CCB [ISB$W_STTM_STAT] = 0;
CCB [ISB$W_FMT [EN]] = 0;
CCB [ISB$A_USER_FP] = 0;
!+
!- Link in previous LUB and make this LUB the current one.
!-
CCB [ISB$A_PREVIOUS_LUB] = .FOR$$A_CUR_LUB;
FOR$$A_CUR_LUB = .CCB;
!+
!- Return with register CCB loaded.
!-
RETURN;
END;

```

! End of routine FOR\$\$CB_PUSH

```

.TITLE FOR$$CB Push, Pop, Allocate, and deallocate LUB
/ISB/RAB
.IDENT \2-005\
.PSECT _FOR$DATA,NOEXE, PIC,2

```

```

00000000 00000 FOR$$A_CUR_LUB::
          .LONG 0
00004 FOR$$A_LUB_TAB::
          .BLKB 512
00204 IOINPROG_VECTOR:
          .BLKB 16
00000000 00000000 00214 INTFIL_QUEUE:
          .LONG 0, 0

```

00000000 0021C V_INTFIL_QUEUE_INIT:
 :LONG 0

FOR\$\$V_IOINPROG= IOINPROG_VECTOR+1
 :EXTRN FOR\$\$ERRSNS_SAV
 :EXTRN FOR\$\$SIG_NO_LUB
 :EXTRN FOR\$\$SIG_DATCOR
 :EXTRN FOR\$\$SIGNAL_STO
 :EXTRN FOR\$\$GET_VM, FOR\$\$FREE_VM

 :PSECT _FOR\$CODE, NOWRT, SHR, PIC, 2

00000077	8F		52	D1	00000	FOR\$\$CB_PUSH::				
			05	14	00007	CMPL	LOGICAL_UNIT, #119	:	0325	
			52	D1	00009	BGTR	1\$:		
	50		06	18	0000C	CMPL	LOGICAL_UNIT, LUN_MIN	:		
			52	DD	0000E	BGEQ	2\$:		
			20	DD	00010	PUSHL	LOGICAL_UNIT	:	0328	
			15	11	00012	PUSHL	#32	:		
15	00000000'	EF	52	E3	00014	BRB	3\$:		
	FFFFFFFFB	8F	52	D1	0001C	BBCS	LOGICAL_UNIT, FOR\$\$V_IOINPROG, 4\$:	0337	
			0C	13	00023	CMPL	LOGICAL_UNIT, #-5	:	0339	
			52	DD	00025	BEQL	4\$:		
			28	DD	00027	PUSHL	LOGICAL_UNIT	:	0342	
			02	FB	00029	PUSHL	#40	:		
00000000G	00		05	00030	CALLS	#2, FOR\$\$SIG_NO_LUB		:	0341	
			09	12	00039	RSB		:	0359	
	5B	00000000'	52	D0	00031	MOVL	FOR\$\$AA_LUB_TAB+32[LOGICAL_UNIT], CCB	:	0365	
			09	12	00039	BNEQ	5\$:	0367	
			52	DD	0003B	PUSHL	LOGICAL_UNIT	:		
0000V	CF		01	FB	0003D	CALLS	#1, ALLOCATE	:		
			12	11	00042	BRB	7\$:		
			52	AB	B1	00044	CMPW	-58(CCB), LOGICAL_UNIT	:	0374
			05	12	00048	BNEQ	6\$:		
			01	6B	91	0004A	CMPB	(CCB), #1	:	0375
			07	13	0004D	BEQL	7\$:		
00000000G	00		00	FB	0004F	CALLS	#0, FOR\$\$SIG_DATCOR	:	0377	
			96	AB	B4	00056	CLRW	-106(CCB)	:	0384
			FF72	CB	B4	00059	CLRW	-142(CCB)	:	0385
			FF4C	CB	D4	0005D	CLRL	-180(CCB)	:	0386
FF48	CB	00000000'	EF	D0	00061	MOVL	FOR\$\$A_CUR_LUB, -184(CCB)	:	0392	
00000000'	EF		5B	D0	0006A	MOVL	CCB, FOR\$\$A_CUR_LUB	:	0393	
			05	00071	RSB			:	0400	

; Routine Size: 114 bytes, Routine Base: _FOR\$CODE + 0000

; 338 0401 1

```
340 0402 1 ROUTINE ALLOCATE (%SBTTL'Allocate CCB'  
341 0403 1 LOGICAL_UNIT : LUN to which to allocate the CCB  
342 0404 1 ) : CALL_CCB NOVALUE = : Allocate LUB/ISB/RAB  
343 0405 1  
344 0406 1 +-  
345 0407 1 FUNCTIONAL DESCRIPTION:  
346 0408 1  
347 0409 1 Allocate heap storage for the LUB/ISB/RAB/FAB/NAM. This is done  
348 0410 1 the first time a logical unit is referenced, and the first  
349 0411 1 time after a CLOSE.  
350 0412 1  
351 0413 1 If this is an ENCODE/DECODE/Internal File, try getting a 'short LUB'  
352 0414 1 from Q_INTFIL_QUEUE. If empty, allocate a short LUB.  
353 0415 1  
354 0416 1 CALLING SEQUENCE:  
355 0417 1  
356 0418 1 ALLOCATE (.LOGICAL_UNIT)  
357 0419 1  
358 0420 1 FORMAL PARAMETERS:  
359 0421 1  
360 0422 1 LOGICAL_UNIT.r.l.v LUN to which to allocate the CCB  
361 0423 1  
362 0424 1 IMPLICIT INPUTS:  
363 0425 1  
364 0426 1 INTFIL_QUEUE Queue of internal file LUBs  
365 0427 1  
366 0428 1 IMPLICIT OUTPUTS  
367 0429 1  
368 0430 1 FOR$AA_LUB_TAB [.LOGICAL_UNIT] and CCB are set  
369 0431 1  
370 0432 1 SIDE EFFECTS:  
371 0433 1  
372 0434 1 Allocates virtual storage.  
373 0435 1 Signals if virtual storage is exhausted.  
374 0436 1  
375 0437 1 --  
376 0438 1  
377 0439 2 BEGIN  
378 0440 2  
379 0441 2 EXTERNAL REGISTER  
380 0442 2 CCB : REF $FOR$CCB_DECL;  
381 0443 2  
382 0444 2 BIND  
383 0445 2 FAB = CCB: REF $FOR$FAB_CCB_STRUCT,  
384 0446 2 NAM = CCB: REF $FOR$NAM_CCB_STRUCT;  
385 0447 2  
386 0448 2 BUILTIN  
387 0449 2 REMQUE;  
388 0450 2  
389 0451 2 +-  
390 0452 2 Split depending on whether or not this is an internal file.  
391 0453 2 -  
392 0454 2  
393 0455 2 IF .LOGICAL_UNIT NEQ LUB$K_LUN_ENCD  
394 0456 2 THEN  
395 0457 3 BEGIN  
396 0458 3
```

```
397 0459      !+
398 0460      ! This is not an internal file or ENCODE/DECODE. Allocate a full-length
399 0461      ! LUB from heap storage and initialize it.
400 0462      !-
401 0463
402 0464      CCB = FOR$$GET VM ((ISB$K_ISB_LEN + LUB$K_LUB_LEN + RAB$C_BLN +
403 0465      FAB$C_BLN + NAM$C_BLN), [LOGICAL UNIT];
404 0466      CH$FILL (0, LUB$K_LUB_LEN + RAB$C_BLN + FAB$C_BLN + NAM$C_BLN,
405 0467      .CCB + ISB$K_ISB_LEN);
406 0468      CCB = .CCB + ISB$K_ISB_LEN + LUB$K_LUB_LEN;
407 0469      CCB [LUB$W_LUN] = [LOGICAL UNIT];
408 0470      CCB [RAB$B_BID] = RAB$C_BID;
409 0471      CCB [RAB$B_BLN] = RAB$C_BLN;
410 0472      FAB [FAB$B_BID] = FAB$C_BID;
411 0473      FAB [FAB$B_BLN] = FAB$C_BLN;
412 0474      NAM [NAM$B_BID] = NAM$C_BID;
413 0475      NAM [NAM$B_BLN] = NAM$C_BLN;
414 0476      CCB [RAB$L_FAB] = FAB [0,0,0,0];
415 0477
416 0478      CCB [RAB$V_TPT] = 1;
417 0479      CCB [RAB$V_RAH] = 1;
418 0480      CCB [RAB$V_WBH] = 1;
419 0481      CCB [RAB$V_LOC] = 1;
420 0482      FOR$$AA_LUB_TAB [LOGICAL UNIT] = .CCB;
421 0483      RETURN;
422 0484      END;
423 0485
424 0486      !+
425 0487      ! This is an internal file or ENCODE/DECODE. First check to see if the
426 0488      ! queue of LUBs has been initialized. If not, initialize it.
427 0489      !-
428 0490
429 0491      IF NOT .V_INTFIL_QUEUE_INIT
430 0492      THEN
431 0493          INITIALIZE_INTFIL_QUEUE ();
432 0494
433 0495      !+
434 0496      ! Try to remove a LUB from the head of the queue. If empty,
435 0497      ! allocate one instead.
436 0498      !-
437 0499
438 0500      IF REMQUE (.INTFIL_QUEUE [0], CCB)
439 0501      THEN
440 0502          BEGIN
441 0503              !+
442 0504              ! Queue was empty. Allocate a short LUB and initialize it.
443 0505              !-
444 0506
445 0507              CCB = FOR$$GET VM ((ISB$K_ISB_LEN + LUB$K_LUB_LEN + RAB$C_BLN),
446 0508              [LOGICAL UNIT];
447 0509              CH$FILL (0, [LUB$K_LUB_LEN + RAB$C_BLN], [CCB + ISB$K_ISB_LEN]);
448 0510              CCB = .CCB + ISB$K_ISB_LEN + LUB$K_LUB_LEN;
449 0511              CCB [LUB$W_LUN] = [LOGICAL UNIT];
450 0512              CCB [RAB$B_BID] = RAB$C_BID;
451 0513              CCB [LUB$V_DEALLOC] = 1;          ! Force "deallocation" on POP
452 0514              END
453 0515      ELSE
```

```

: 454      0516 2      CCB = .CCB + ISB$K_ISB_LEN + LUB$K_LUB_LEN;      ! Get right base for CCB
: 455      0517 2
: 456      0518 2      RETURN;      ! With LUB address in CCB
: 457      0519 1      END;

```

00FC 00000 ALLOCATE:

```

.WORD      Save R2,R3,R4,R5,R6,R7      : 0402
MOVAB      FOR$$GET_VM, R7
MOVAB      FOR$$AA_LUB_TAB+32, R6
CMLP      LOGICAL_UNIT, #-5      : 0455
BEQL      1$
PUSHL     LOGICAL_UNIT      : 0465
MOVZWL     #532, -7(SP)      : 0464
CALLS     #2, FOR$$GET_VM
MOVL      R0, CCB
MOVCS     #0, (SP), #0, #344, 188(CCB)      : 0467
MOVAB      288(R11), CCB      : 0468
MOVW      LOGICAL_UNIT, -58(CCB)      : 0469
MOVW      #17409, -(CCB)      : 0470
MOVW      #20483, 68(CCB)      : 0472
MOVW      #24578, 148(CCB)      : 0474
MOVAB     68(CCB), 60(CCB)      : 0476
BISL2     #67074, 4(CCB)      : 0481
MOVL     LOGICAL_UNIT, R0      : 0482
MOVL     CCB, FOR$$AA_LUB_TAB+32[R0]
RET      : 0457
BLBS     V INTFIL_QUEUE_INIT, 2$      : 0491
CALLS     #0, INITIALIZE_INTFIL_QUEUE      : 0493
REMQUE    @INTFIL_QUEUE, CCB      : 0500
BVC      3$
PUSHL     LOGICAL_UNIT      : 0508
MOVZWL     #356, -7(SP)      : 0507
CALLS     #2, FOR$$GET_VM
MOVL      R0, CCB
MOVCS     #0, (SP), #0, #168, 188(CCB)      : 0509
MOVAB     288(R11), CCB      : 0510
MOVW      LOGICAL_UNIT, -58(CCB)      : 0511
MOVW      #1, (CCB)      : 0512
BISB2     #16, -1(CCB)      : 0513
RET      : 0500
MOVAB     288(R11), CCB      : 0516
RET      : 0519

```

; Routine Size: 165 bytes, Routine Base: _FOR\$CODE + 0072

; 458 0520 1

```
460 0521 1 GLOBAL ROUTINE FOR$$CB POP      %SBTTL'Pop current CCB'
461 0522 1   : JSB_CB_POP NOVALOE =
462 0523 1
463 0524 1  +-
464 0525 1  FUNCTIONAL DESCRIPTION:
465 0526 1
466 0527 1      FOR$$CB_POP pops the curenst LUB/ISB/RAB and restores the
467 0528 1      previous pushed down LUB/ISB/RAB, if any (usually none).
468 0529 1      Flags old current LUB/ISB/RAB as no longer having as active I/O statement
469 0530 1
470 0531 1  CALLING SEQUENCE:
471 0532 1
472 0533 1      JSB FOR$$CB_POP ( )
473 0534 1
474 0535 1  FORMAL PARAMETERS:
475 0536 1
476 0537 1      NONE
477 0538 1
478 0539 1  IMPLICIT INPUTS:
479 0540 1
480 0541 1      CCB                      Adr. of current LUB/ISB/RAB
481 0542 1
482 0543 1  IMPLICIT OUTPUTS:
483 0544 1
484 0545 1      CCB                      Set to 0 (to catch attempt to reference after a pop).
485 0546 1
486 0547 1  RETURN VALUE:
487 0548 1
488 0549 1      NONE
489 0550 1
490 0551 1  SIDE EFFECTS:
491 0552 1
492 0553 1      Changes entire I/O system to another logical unit or none at all
493 0554 1      SIGNAL_STOPs FORTRAN INTERNAL ERROR if CB was not active.
494 0555 1  --
495 0556 1
496 0557 2  BEGIN
497 0558 2
498 0559 2  BUILTIN
499 0560 2      TESTBITCC;
500 0561 2
501 0562 2  EXTERNAL REGISTER
502 0563 2      CCB : REF $FOR$CCB_DECL;
503 0564 2
504 0565 2  LOCAL
505 0566 2      LOGICAL_UNIT;
506 0567 2
507 0568 2  +-
508 0569 2  | Pop this CCB.
509 0570 2  |
510 0571 2
511 0572 2  LOGICAL_UNIT = .CCB [LUB$W LUN];
512 0573 2  FOR$$A_CUR_LUB = .CCB [ISB$A_PREVIOUS_LUB];
513 0574 2
514 0575 2  +-
515 0576 2  | Deallocate run-time format
516 0577 2  |
```

```

517 0578 2
518 0579 2
519 0580 2
520 0581 2
521 0582 2
522 0583 2
523 0584 2
524 0585 2
525 0586 2
526 0587 2
527 0588 2
528 0589 2
529 0590 2
530 0591 2
531 0592 2
532 0593 2
533 0594 2
534 0595 2
535 0596 2
536 0597 2
537 0598 2
538 0599 2
539 0600 2
540 0601 2
541 0602 2
542 0603 2
543 0604 2
544 0605 2
545 0606 2
546 0607 2
547 0608 2
548 0609 2
549 0610 2
550 0611 2
551 0612 1

```

```

IF (.CCB [ISB$W_FMT_LEN] NEQ 0)
THEN
  BEGIN
    FOR$$FREE_VM (.CCB [ISB$W_FMT_LEN], .CCB [ISB$A_FMT_BEG]);
    CCB [ISB$W_FMT_LEN] = 0;
    CCB [ISB$A_FMT_BEG] = 0;
  END;

  +
  Deallocate this LUB if requested to.
  -

IF (.CCB [LUB$V_DEALLOC])
THEN
  DEALLOCATE (.LOGICAL_UNIT);

  +
  Flag old current LUB/ISB/RAB as no longer having
  an I/O statement in progress.
  If LUB was not active, then signal OTSS_INTDATCOR (INTERNAL DATA
  CORRUPTED IN RUN-TIME LIBRARY).
  -

IF (TESTBITCC (FOR$$V_IOINPROG [.LOGICAL_UNIT]))
THEN
  IF .LOGICAL_UNIT NEQU LUB$K_LUN_ENCD
  THEN
    FOR$$SIG_DATCOR ();

CCB = 0;

RETURN;

END;

```

! End of FOR\$\$CB_POP routine

	7E	C6	AB	32	00000	FOR\$\$CB_POP::		
						CVTWL	-58(CCB), LOGICAL_UNIT	: 0572
	00000000'	EF	FF48	CB	D0 00004	MOVL	-184(CCB), FOR\$\$A_CUR_LUB	: 0573
		50	FF72	CB	3C 0000D	MOVZWL	-142(CCB), R0	: 0579
				15	13 00012	BEQL	1\$	
			FF7C	CB	DD 00014	PUSHL	-132(CCB)	: 0582
				50	DD 00018	PUSHL	R0	
	00000000G	00		02	FB 0001A	CALLS	#2, FOR\$\$FREE_VM	
			FF72	CB	B4 00021	CLRW	-142(CCB)	: 0583
			FF7C	CB	D4 00025	CLRL	-132(CCB)	: 0584
07	FF	AB		04	E1 00029	1\$: BBC	#4, -1(CCB), 2\$: 0591
				6E	DD 0002E	PUSHL	LOGICAL_UNIT	: 0593
	0000V	CF		01	FB 00030	CALLS	#1, DEALLOCATE	
10	00000000'	EF		6E	E4 00035	2\$: BBSC	LOGICAL_UNIT, FOR\$\$V_IOINPROG, 3\$: 0602
	FFFFFFFFB	8F		6E	D1 0003D	CMPL	LOGICAL_UNIT, #-5	: 0604
				07	13 00044	BEQL	3\$	
	00000000G	00		00	FB 00046	CALLS	#0, FOR\$\$SIG_DATCOR	: 0606

FOR\$\$CB
2-005

Push, Pop, Allocate, and deallocate LUB/ISB/RAB
Pop current CCB

F 8
16-Sep-1984 00:13:56
14-Sep-1984 12:31:38

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[FORRTL.SRC]FORCB.B32;1 Page 14 (5)

SE

5B D4 0004D 3\$:
04 C0 0004F
05 00052

CLRL CCB
ADDL2 #4, SP
RSB

: 0608
: 0612
:

; Routine Size: 83 bytes, Routine Base: _FOR\$CODE + 0117

; 552 0613 1


```
0614 1 ROUTINE DEALLOCATE (%SBTTL'Deallocate a CCB'
0615 1 LOGICAL UNIT ! The LUN on which to deallocate
0616 1 ) : CALL_CCB NOVALUE =
0617 1
0618 1 !++
0619 1 FUNCTIONAL DESCRIPTION:
0620 1
0621 1 Release the heap storage associated with a CCB. This is done after
0622 1 a CLOSE. If the file is an internal file, insert the LUB on
0623 1 INTFIL_QUEUE rather than deallocating it.
0624 1
0625 1 CALLING SEQUENCE:
0626 1
0627 1 DEALLOCATE (.LOGICAL_UNIT)
0628 1
0629 1 FORMAL PARAMETERS:
0630 1
0631 1 LOGICAL_UNIT.rl.v The LUN for which to deallocate the CCB
0632 1
0633 1 IMPLICIT INPUTS:
0634 1
0635 1 INTFIL_QUEUE
0636 1 Several fields of the LUB
0637 1
0638 1 IMPLICIT OUTPUTS:
0639 1
0640 1 INTFIL_QUEUE
0641 1 FOR$A_LUB_TAB [.LOGICAL_UNIT] is cleared
0642 1
0643 1 SIDE EFFECTS:
0644 1
0645 1 Deallocates heap storage
0646 1
0647 1 --
0648 1
0649 2 BEGIN
0650 2
0651 2 BUILTIN
0652 2 INSQUE
0653 2 TESTBITCC;
0654 2
0655 2 EXTERNAL REGISTER
0656 2 CCB : REF $FOR$CCB_DECL;
0657 2
0658 2 !+
0659 2 ! Split depending on whether or not this is an internal file/ENCODE/DECODE.
0660 2 !-
0661 2
0662 2 IF .CCB [LUB$W_LUN] NEQ LUB$K_LUN_ENCD
0663 2 THEN
0664 2 BEGIN
0665 2
0666 2 !+
0667 2 ! Remove this LUB from the LUB table.
0668 2 !-
0669 2
0670 2 FOR$AA_LUB_TAB [.LOGICAL_UNIT] = 0;
```

```
611 0671
612 0672
613 0673
614 0674
615 0675
616 0676
617 0677
618 0678
619 0679
620 0680
621 0681
622 0682
623 0683
624 0684
625 0685
626 0686
627 0687
628 0688
629 0689
630 0690
631 0691
632 0692
633 0693
634 0694
635 0695
636 0696
637 0697
638 0698
639 0699
640 0700
641 0701
642 0702
643 0703
644 0704
645 0705
646 0706
647 0707
648 0708
649 0709
650 0710
651 0711
652 0712
653 0713
654 0714
655 0715
656 0716
657 0717
658 0718
659 0719
660 0720
661 0721
662 0722
663 0723
664 0724
665 0725
666 0726
667 0727

+
- Deallocate record buffer, if present.
-
IF (( NOT .CCB [LUB$V_USER_RBUF]) AND (.CCB [LUB$A_UBF] NEQA 0))
THEN
FOR$$FREE_VM (.CCB [LUB$W_RBUF_SIZE], .CCB [LUB$A_UBF]);

+
- Deallocate FAB if allocated by ASSIGN/FDBSET. If filename
also allocated, deallocate it.
-
IF .CCB [LUB$A_FAB] NEQA 0
THEN
BEGIN
LOCAL
HEAP_FAB: REF BLOCK [, BYTE];
HEAP_FAB = .CCB [LUB$A_FAB];
IF .HEAP_FAB [FAB$B_FNS] NEQU 0
THEN
FOR$$FREE_VM (.HEAP_FAB [FAB$B_FNS], .HEAP_FAB [FAB$L_FNA]);
FOR$$FREE_VM (.HEAP_FAB [FAB$B_BLN], .HEAP_FAB);
END;

+
- Deallocate resultant name string, if present.
-
IF (.CCB [LUB$V_VIRT_RSN])
THEN
FOR$$FREE_VM (.CCB [LUB$B_RSL], .CCB [LUB$A_RSN]);

+
- Deallocate RFA cache, if present.
-
IF .CCB [LUB$A_RFA_CACHE_BEG] NEQA 0
THEN
FOR$$FREE_VM ((RCE_K_CACHE_SIZE * RCE_S_RCE_STRUCT),
.CCB [LUB$A_RFA_CACHE_BEG]);

+
- Deallocate LJB memory.
-
FOR$$FREE_VM ((ISB$K_ISB_LEN + LUB$K_LUB_LEN + RAB$C_BLN +
FAB$C_BLN + NAM$C_BLN), .CCB - (ISB$R_ISB_LEN + [LUB$K_LUB_LEN]));

RETURN;
END;

+
- This is an ENCODE/DECODE/internal file. Insert the LUB on the queue.
Use the first two longwords of the ISB as the queue link.
-
```

: 668 0728 2
: 669 0729 2
: 670 0730 2
: 671 0731 2
: 672 0732 2
: 673 0733 1

INSQUE (.CCB - (ISB\$K_ISB_LEN + LUB\$K_LUB_LEN), INTFIL_QUEUE);
RETURN;
END;

000C 00000 DEALLOCATE:

					.WORD	Save R2,R3		: 0614	
FFF8	53	00000000G	00	9E	00002	MOVAB	FOR\$\$FREE_VM, R3		
	8F	C6	AB	B1	00009	CMPW	-58(CCB), #-5	: 0662	
			6B	13	0000F	BEQL	6\$		
	50	04	AC	D0	00011	MOVL	LOGICAL_UNIT, R0	: 0670	
		00000000'	EF40	D4	00015	CLRL	FOR\$\$AA_LUB_TAB+32[R0]		
			FF	AB	95	0001C	TSTB	-1(CCB)	: 0676
				OF	19	0001F	BLSS	1\$	
			9C	AB	D5	00021	TSTL	-100(CCB)	
				0A	13	00024	BEQL	1\$	
			9C	AB	DD	00026	PUSHL	-100(CCB)	: 0678
7E	D2		AB	3C	00029	MOVZWL	-46(CCB), -(SP)		
63			02	FB	0002D	CALLS	#2, FOR\$\$FREE_VM		
			E8	AB	D5	00030	TSTL	-24(CCB)	: 0685
				1C	13	00033	BEQL	3\$	
	52		E8	AB	D0	00035	MOVL	-24(CCB), HEAP_FAB	: 0690
			34	A2	95	00039	TSTB	52(HEAP_FAB)	: 0691
				0A	13	0003C	BEQL	2\$	
			2C	A2	DD	0003E	PUSHL	44(HEAP_FAB)	: 0693
7E	34		A2	9A	00041	MOVZBL	52(HEAP_FAB), -(SP)		
63			02	FB	00045	CALLS	#2, FOR\$\$FREE_VM		
			52	DD	00048	PUSHL	HEAP_FAB	: 0694	
7E	01		A2	9A	0004A	MOVZBL	1(HEAP_FAB), -(SP)		
63			02	FB	0004E	CALLS	#2, FOR\$\$FREE_VM		
0A	FE		AB	E9	00051	BLBC	-2(CCB), 4\$: 0701	
			F8	AB	DD	00055	PUSHL	-8(CCB)	: 0703
7E	F7		AB	9A	00058	MOVZBL	-9(CCB), -(SP)		
63			02	FB	0005C	CALLS	#2, FOR\$\$FREE_VM		
			C8	AB	D5	0005F	TSTL	-56(CCB)	: 0709
				0B	13	00062	BEQL	5\$	
			C8	AB	DD	00064	PUSHL	-56(CCB)	: 0712
7E	0190		8F	3C	00067	MOVZWL	#400, -(SP)	: 0711	
63			02	FB	0006C	CALLS	#2, FOR\$\$FREE_VM		
		FEE0	CB	9F	0006F	PUSHAB	-288(CCB)	: 0719	
7E	0214		8F	3C	00073	MOVZWL	#532, -(SP)	: 0718	
63			02	FB	00078	CALLS	#2, FOR\$\$FREE_VM		
				04	0007B	RET		: 0664	
		00000000'	EF	FEE0	CB	0E	0007C	6\$: INSQUE -288(CCB), INTFIL_QUEUE	: 0729
				04	00085	RET		: 0733	

: Routine Size: 134 bytes, Routine Base: _FOR\$CODE + 016A

: 674 0734 1

```

: 676 0735 1 GLOBAL ROUTINE FOR$$CB_GET %SBTTL'GET current CCB'
: 677 0736 1 : JSB_CB_GET NOVALDE =
: 678 0737 1
: 679 0738 1 ++
: 680 0739 1 FUNCTIONAL DESCRIPTION:
: 681 0740 1
: 682 0741 1 FOR$$CB_GET gets the curenst LUB/ISB/RAB.
: 683 0742 1 This routine is only called from non-shared procedures which
: 684 0743 1 can't access FOR$$A_CUR_LUB directly. (Entry vectors for
: 685 0744 1 data would mean that the code would have to change when the
: 686 0745 1 decision to make a module shared or non-shared is changed.
: 687 0746 1 Unless the LINKER got smarter and changed the level of indirection
: 688 0747 1 on data references which were vectored.)
: 689 0748 1
: 690 0749 1 CALLING SEQUENCE:
: 691 0750 1
: 692 0751 1 JSB FOR$$CB_GET ( )
: 693 0752 1
: 694 0753 1 FORMAL PARAMETERS:
: 695 0754 1
: 696 0755 1 NONE
: 697 0756 1
: 698 0757 1 IMPLICIT INPUTS:
: 699 0758 1
: 700 0759 1 FOR$$A_CUR_LUB Adr. of current LUB/ISB/RAB
: 701 0760 1
: 702 0761 1 IMPLICIT OUTPUTS.
: 703 0762 1
: 704 0763 1 CCB Set to adr. of current LUB/ISB/RAB.
: 705 0764 1
: 706 0765 1 RETURN VALUE:
: 707 0766 1
: 708 0767 1 NONE
: 709 0768 1
: 710 0769 1 SIDE EFFECTS:
: 711 0770 1
: 712 0771 1 NONE
: 713 0772 1 --
: 714 0773 1
: 715 0774 2 BEGIN
: 716 0775 2
: 717 0776 2 EXTERNAL REGISTER
: 718 0777 2 CCB : REF $FOR$CCB_DECL;
: 719 0778 2
: 720 0779 2 CCB = .FOR$$A_CUR_LUB;
: 721 0780 2
: 722 0781 2 RETURN
: 723 0782 2
: 724 0783 1 END; ! End of FOR$$CB_GET routine

```

```

5B 00000000' EF D0 0000 FOR$$CB_GET::
                                MOVL FOR$$A_CUR_LUB, CCB
                                RSB

```

```

: 0779
: 0783

```

FOR\$CB
2-005

Push, Pop, Allocate, and deallocate LUB/ISB/RAB
GET current CCB

K 8
16-Sep-1984 00:13:56
14-Sep-1984 12:31:38

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[FORRTL.SRC]FORCB.B32;1
Page 19
(7)

; Routine Size: 8 bytes, Routine Base: _FOR\$CODE + 01F0

; 725 0784 1

```

: 727 0785 1 GLOBAL ROUTINE FOR$$CB_FETCH (%SBTTL'Fetch a LUB, or 0'
: 728 0786 1     LUN                                ! LUN of the LUB
: 729 0787 1     ) : CALL_CCB NOVALUE =
: 730 0788 1
: 731 0789 1 !++
: 732 0790 1 ! FUNCTIONAL DESCRIPTION:
: 733 0791 1
: 734 0792 1     FOR$$CB_FETCH returns the CCB address for a given LUN without
: 735 0793 1     "pushing" it. This is used by FOR$$CLOSE_ALL and FOR$INQUIRE.
: 736 0794 1     ASTs must be disabled before FOR$$CB_FETCH is called and not
: 737 0795 1     reenabled until after the CCB is no longer needed.
: 738 0796 1
: 739 0797 1 ! CALLING SEQUENCE:
: 740 0798 1
: 741 0799 1     CALL FOR$$CB_FETCH (LUN)
: 742 0800 1
: 743 0801 1 ! FORMAL PARAMETERS:
: 744 0802 1
: 745 0803 1     LUN.rl.v                Logical Unit Number at which to "peek"
: 746 0804 1
: 747 0805 1 ! IMPLICIT INPUTS:
: 748 0806 1
: 749 0807 1     FOR$$V LUN_OWNR          Table of LUN owners
: 750 0808 1     FOR$$AA_LUB_TAB         Table of pointers to LUBs
: 751 0809 1
: 752 0810 1 ! IMPLICIT OUTPUTS:
: 753 0811 1
: 754 0812 1     CCB                    This register is set to 0 if the LUN is not owned by FORTRAN
: 755 0813 1     or is not allocated, or to the address of the 'UB/ISB/RAB
: 756 0814 1     otherwise.
: 757 0815 1
: 758 0816 1 ! RETURN VALUE:
: 759 0817 1
: 760 0818 1     NONE
: 761 0819 1
: 762 0820 1 ! SIDE EFFECTS:
: 763 0821 1
: 764 0822 1     NONE
: 765 0823 1 !--
: 766 0824 1
: 767 0825 2     BEGIN
: 768 0826 2
: 769 0827 2     EXTERNAL REGISTER
: 770 0828 2     CCB : REF $FOR$CCB_DECL;
: 771 0829 2
: 772 0830 2     CCB = .FOR$$AA_LUB_TAB [.LUN];
: 773 0831 2
: 774 0832 2     RETURN;
: 775 0833 1     END;

```

! of routine FOR\$\$CB_FETCH

```

0000 0000
50      04  AC  DO 00002
5B 00000000'EF40  DO 00006

```

```

.ENTRY FOR$$CB_FETCH, Save nothing
MOVL  LUN, R0
MOVL  FOR$$AA_LUB_TAB+32[R0], CCB

```

```

: 0785
: 0830
:

```



```

: 778 0835 1 GLOBAL ROUTINE FOR$$NEXT_LUN (%SBTTL'Get next LUN which might be open'
: 779 0836 1     FLAG: REF VECTOR [ , LONG],           ! First-time and last-time flag
: 780 0837 1     LUN: REF VECTOR [ , LONG]           ! Logical Unit Number
: 781 0838 1     ) : NOVALUE =
: 782 0839 1
: 783 0840 1     ++
: 784 0841 1     FUNCTIONAL DESCRIPTION:
: 785 0842 1
: 786 0843 1         FOR$$NEXT_LUN gets a LUN which might be open.  It is used by
: 787 0844 1         the exit handler declared by FORTRAN OPEN, which must look
: 788 0845 1         through all the LUNs and do the DELETE or PRINT handling by
: 789 0846 1         calling CLOSE.  (RMS close won't do DELETE or PRINT handling.)
: 790 0847 1         This routine scans the table of LUB pointers and returns those
: 791 0848 1         which are non-zero.  The caller must use CB_PUSH and CB_POP
: 792 0849 1         to obtain control of the LUB.
: 793 0850 1
: 794 0851 1     CALLING SEQUENCE:
: 795 0852 1
: 796 0853 1         CALL FOR$$NEXT_LUN (FLAG, LUN)
: 797 0854 1
: 798 0855 1     FORMAL PARAMETERS:
: 799 0856 1
: 800 0857 1         FLAG.mv.r                       If 0 on entry, this is the first call
: 801 0858 1                                         and LUN is invalid.  If 1 on entry, LUN
: 802 0859 1                                         is the last LUN processed.  On exit, 0
: 803 0860 1                                         means that there are no more LUNs, and 1
: 804 0861 1                                         means that LUN contains the Logical Unit
: 805 0862 1                                         Number to process.
: 806 0863 1         LUN.ml.r                       Logical Unit Number, as described above.
: 807 0864 1
: 808 0865 1     IMPLICIT INPUTS:
: 809 0866 1
: 810 0867 1         FOR$$AA_LUB_TAB
: 811 0868 1
: 812 0869 1     IMPLICIT OUTPUTS:
: 813 0870 1
: 814 0871 1         NONE
: 815 0872 1
: 816 0873 1     RETURN VALUE:
: 817 0874 1
: 818 0875 1         NONE
: 819 0876 1
: 820 0877 1     SIDE EFFECTS:
: 821 0878 1
: 822 0879 1         NONE
: 823 0880 1     --
: 824 0881 1
: 825 0882 2     BEGIN
: 826 0883 2
: 827 0884 2     LOCAL
: 828 0885 2         LOCAL_LUN;
: 829 0886 2
: 830 0887 2     !+
: 831 0888 2     ! If this is the first entry, arrange to return the first logical
: 832 0889 2     ! unit.
: 833 0890 2     !-
: 834 0891 2
```



```

835 0892 2 IF NOT .FLAG [0]
836 0893 THEN
837 0894 BEGIN
838 0895 FLAG [0] = 1;
839 0896 LOCAL_LUN = LUB$K_ILUN_MIN;
840 0897 END
841 0898 ELSE
842 0899 BEGIN
843 0900 LOCAL_LUN = .LUN [0] + 1;
844 0901 END;
845 0902
846 0903 !+
847 0904 ! While the unit number is in range, look for a LUB entry that is
848 0905 ! non-zero.
849 0906 !-
850 0907
851 0908 WHILE (.LOCAL_LUN LEQ LUB$K_LUN_MAX) DO
852 0909 BEGIN
853 0910 IF .FOR$$AA_LUB_TAB [.LOCAL_LUN] NEQ 0
854 0911 THEN
855 0912 BEGIN
856 0913 LUN [0] = .LOCAL_LUN;
857 0914 RETURN;
858 0915 END;
859 0916 LOCAL_LUN = .LOCAL_LUN + 1;
860 0917 END;
861 0918
862 0919 !+
863 0920 ! We dropped out of the loop. Return failure.
864 0921 !-
865 0922
866 0923 FLAG [0] = 0;
867 0924
868 0925 RETURN;
869 0926 END;

```

! End of FOR\$\$NEXT_LUN routine

				0000	00000	.ENTRY	FOR\$\$NEXT_LUN, Save nothing	:	0835
			04	BC	E8	BLBS	@FLAG, 1\$:	0892
	04			01	D0	MOVL	#1, @FLAG	:	0895
				08	CE	MNEGL	#8, LOCAL_LUN	:	0896
				05	11	BRB	2\$:	0892
				01	C1	ADDL3	#1, @LUN, LOCAL_LUN	:	0900
50	08			50	D1	CMPL	LOCAL_LUN, #119	:	0908
	00000077			12	14	BGTR	4\$:	
				40	D5	TSTL	FOR\$\$AA_LUB_TAB+32[LOCAL_LUN]	:	0910
				05	13	BEQL	3\$:	
	08	BC		50	D0	MOVL	LOCAL_LUN, @LUN	:	0913
				04	0002A	RET		:	0912
				50	D6	INCL	LOCAL_LUN	:	0916
				E5	11	BRB	2\$:	0908
			04	BC	D4	CLRL	@FLAG	:	0923
				04	00032	RET		:	0926

FOR\$CB
2-005

Push, Pop, Allocate, and deallocate LUB/ISB/RAB
Get next LUN which might be open

16-Sep-1984 00:13:56
14-Sep-1984 12:31:38

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[FORRTL.SRC]FORCB.B32;1 (9)

Page 24

FC
1-

; Routine Size: 51 bytes, Routine Base: _FOR\$CODE + 0207

; 870 0927 1

```

872 0928 1 %SBTTL'FOR$$FP_MATCH - Find current incarnation'
873 0929 1 GLOBAL ROUTINE FOR$$FP_MATCH (
874 0930 1     SIG_FP
875 0931 1     ) : CALL_CCB NOVALUE =
876 0932 1
877 0933 1
878 0934 1 ++
879 0935 1 FUNCTIONAL DESCRIPTION:
880 0936 1     FOR$$FP_MATCH is part of the I/O in progress handling scheme.
881 0937 1     It is called with one argument, the value of the frame pointer
882 0938 1     desired. It looks through the current ISB chain until it finds
883 0939 1     an ISB that has the desired FP in ISB$A_USER_FP. This means that
884 0940 1     that ISB was the one in effect when the I/O in progress handler
885 0941 1     was established. If it finds one, external register CCB is set
886 0942 1     to the CCB of that ISB. If no match is found, there is something
887 0943 1     seriously wrong in the database so error OTSS_INTDATCOR is
888 0944 1     signalled.
889 0945 1
890 0946 1 CALLING SEQUENCE:
891 0947 1
892 0948 1     CALL FOR$$FP_MATCH (SIG_FP)
893 0949 1
894 0950 1 FORMAL PARAMETERS:
895 0951 1
896 0952 1     SIG_FP.r.l.v
897 0953 1     The FP present in the signal mechanism
898 0954 1     list when the I/O in progress handler
899 0955 1     was signalled. This value is searched for
900 0956 1     in the current ISB chain.
901 0957 1 IMPLICIT INPUTS:
902 0958 1
903 0959 1     FOR$$AA_LUB_TAB
904 0960 1     FOR$$A_CUR_LUB
905 0961 1
906 0962 1 IMPLICIT OUTPUTS:
907 0963 1
908 0964 1     CCB
909 0965 1     This register is set to the address of the
910 0966 1     ISB/LUB/RAB block that has SIG_FP in its
911 0967 1     ISB$A_USER_FP.
912 0968 1 RETURN VALUE:
913 0969 1
914 0970 1     NONE
915 0971 1
916 0972 1 SIDE EFFECTS:
917 0973 1
918 0974 1     Signals OTSS_INTDATCOR (Internal data corrupted in Run-Time Library)
919 0975 1     if no ISB is found that matches SIG_FP.
920 0976 1 --
921 0977 1
922 0978 2 BEGIN
923 0979 2
924 0980 2 EXTERNAL REGISTER
925 0981 2     CCB : REF $FOR$CCB_DECL;
926 0982 2
927 0983 2 LOCAL
928 0984 2     LOGICAL_UNIT;

```

! Logical unit number of current LUB

```

: 929 0985 2
: 930 0986 2
: 931 0987 2 | + Get current LUB
: 932 0988 2 | -
: 933 0989 2
: 934 0990 2 CCB = .FOR$$A_CUR_LUB;
: 935 0991 2
: 936 0992 2 | +
: 937 0993 2 | Search through ISB chain to find matching FP
: 938 0994 2 | -
: 939 0995 2
: 940 0996 2 WHILE .CCB NEQ 0 DO
: 941 0997 2 BEGIN
: 942 0998 2 LOGICAL_UNIT = .CCB [LUB$W_LUN];
: 943 0999 2
: 944 1000 2 IF .CCB [ISB$A_USER_FP] EQL .SIG_FP
: 945 1001 2 THEN
: 946 1002 2 RETURN;
: 947 1003 2
: 948 1004 2 CCB = .CCB [ISB$A_PREVIOUS_LUB];
: 949 1005 2 END;
: 950 1006 2
: 951 1007 2 | +
: 952 1008 2 | If we get here, then there must not have been a match.
: 953 1009 2 | This should never happen, therefore signal an error.
: 954 1010 2 | -
: 955 1011 2
: 956 1012 2 FOR$$SIG_DATCOR ();
: 957 1013 2 RETURN;
: 958 1014 2 END;

```

			0000 00000	.ENTRY	FOR\$\$FP_MATCH, Save nothing	:	0929
	5B	00000000'	EF D0 00002	MOVL	FOR\$\$A_CUR_LUB, CCB	:	0990
			13 13 00009 1\$:	BEQL	2\$:	0996
	50	C6	AB 32 0000B	CVTWL	-58(CCB), LOGICAL_UNIT	:	0998
04	AC	FF4C	CB D1 0000F	CML	-180(CCB), SIG_FP	:	1000
			0E 13 00015	BEQL	3\$:	
	5B	FF48	CB D0 00017	MOVL	-184(CCB), CCB	:	1004
			EB 11 0001C	BRB	1\$:	0996
00000000G	00		00 FB 0001E 2\$:	CALLS	#0, FOR\$\$SIG_DATCOR	:	1012
			04 00025 3\$:	RET		:	1014

; Routine Size: 38 bytes, Routine Base: _FOR\$CODE + 023A

```

: 960 1015 1 %SBTTL 'INITIALIZE_INTFIL_QUEUE - Initialize INTFIL_QUEUE'
: 961 1016 1 ROUTINE INITIALIZE_INTFIL_QUEUE
: 962 1017 1 : NOVALUE =
: 963 1018 1
: 964 1019 1 !++
: 965 1020 1 FUNCTIONAL DESCRIPTION:
: 966 1021 1
: 967 1022 1     Initializes INTFIL_QUEUE to be an empty queue.
: 968 1023 1
: 969 1024 1 CALLING SEQUENCE:
: 970 1025 1
: 971 1026 1     INITIALIZE_INTFIL_QUEUE ( )
: 972 1027 1
: 973 1028 1 FORMAL PARAMETERS:
: 974 1029 1
: 975 1030 1     NONE
: 976 1031 1
: 977 1032 1 IMPLICIT INPUTS:
: 978 1033 1
: 979 1034 1     INTFIL_QUEUE
: 980 1035 1     V_INTFIL_QUEUE_INIT
: 981 1036 1
: 982 1037 1 IMPLICIT OUTPUTS:
: 983 1038 1
: 984 1039 1     INTFIL_QUEUE
: 985 1040 1     V_INTFIL_QUEUE_INIT
: 986 1041 1
: 987 1042 1 COMPLETION STATUS:
: 988 1043 1
: 989 1044 1     NONE
: 990 1045 1
: 991 1046 1 SIDE EFFECTS:
: 992 1047 1
: 993 1048 1     Makes INTFIL_QUEUE an empty queue.
: 994 1049 1
: 995 1050 1 SIGNALLED ERRORS:
: 996 1051 1
: 997 1052 1     NONE
: 998 1053 1 --
: 999 1054 1
1000 1055 2 BEGIN
1001 1056 2
1002 1057 2 LOCAL
1003 1058 2     AST_STATUS;                ! Previous AST enable status
1004 1059 2
1005 1060 2 BUILTIN
1006 1061 2     TESTBITCS;
1007 1062 2
1008 1063 2 !+
1009 1064 2 ! Disable ASTs.
1010 1065 2 !-
1011 1066 2
1012 1067 2 AST_STATUS = $SETAST (ENBFLG = 0);
1013 1068 2
1014 1069 2 !+
1015 1070 2 ! If V_INTFIL_QUEUE_INIT is still clear, initialize INTFIL_QUEUE to
: 1016 1071 2 ! be an empty queue. Set V_INTFIL_QUEUE_INIT.

```

```

: 1017      1072  2      !-
: 1018      1073  2
: 1019      1074  2      IF TESTBITCS (V_INTFIL_QUEUE_INIT)
: 1020      1075  2      THEN
: 1021      1076  2          BEGIN
: 1022      1077  3          INTFIL_QUEUE [0] = INTFIL_QUEUE;      ! Set forward link
: 1023      1078  3          INTFIL_QUEUE [1] = .INTFIL_QUEUE [0]; ! Set backward link
: 1024      1079  2          END;
: 1025      1080  2
: 1026      1081  2      !+
: 1027      1082  2      ! Reenable ASTs if previously enabled.
: 1028      1083  2      !-
: 1029      1084  2
: 1030      1085  2      IF .AST_STATUS EQL SS$_WASSET
: 1031      1086  2      THEN
: 1032      1087  2          $SETAST (ENBFLG = 1);
: 1033      1088  2
: 1034      1089  2      RETURN;
: 1035      1090  2
: 1036      1091  1      END;

```

! End of routine INITIALIZE_INTFIL_QUEUE

.EXTRN SYS\$SETAST

		000C 0000	INITIALIZE_INTFIL_QUEUE:			
		53 00000000G	00 9E 00002	.WORD	Save R2,R3	: 1016
		52 00000000'	EF 9E 00009	MOVAB	SYS\$SETAST, R3	
			7E D4 00010	MOVAB	INTFIL_QUEUE, R2	
		63	01 FB 00012	CLRL	-(SP)	: 1067
07	08	A2	00 E2 00015	CALLS	#1, SYS\$SETAST	
		62	62 9E 0001A	BBSS	#0, V_INTFIL_QUEUE_INIT, 1\$: 1074
	04	A2	62 D0 0001D	MOVAB	INTFIL_QUEUE, INTFIL_QUEUE	: 1077
		09	50 D1 00021 1\$:	MOVL	INTFIL_QUEUE, INTFIL_QUEUE+4	: 1078
			05 12 00024	CML	AST_STATUS, #9	: 1085
			01 DD 00026	BNEQ	2\$	
		63	01 FB 00028	PUSHL	#1	: 1087
			04 0002B 2\$:	CALLS	#1, SYS\$SETAST	
				RET		: 1091

: Routine Size: 44 bytes, Routine Base: _FOR\$CODE + 0260

: 1038 1092 1 END ! End of module FOR\$\$CB
: 1039 1093 1
: 1040 1094 0 ELUDOM

FOR\$\$CB_RET== FOR\$\$CB_POP

PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$DATA	544	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_FOR\$CODE	652	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	23	0	581	00:01.0
_\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	192	27	52	00:00.5
_\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

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

: Size: 652 code + 544 data bytes
: Run Time: 00:17.3
: Elapsed Time: 00:43.8
: Lines/CPU Min: 3794
: Lexemes/CPU-Min: 14184
: Memory Used: 117 pages
: Compilation Complete

COMR50WD
LIS

FORDATEDS
LIS

FORDECOMO
LIS

FORB
LIS

COMSETST
LIS

FORASSOC
LIS

FORCLOSEF
LIS

FORDATE
LIS

FORCLOSE
LIS

FORDECOMP
LIS

FORDELETE
LIS

COMRAD50
LIS

COMUSEREX
LIS

FORBITOPS
LIS

FORDEFINE
LIS

FORBACKSP
LIS

FORDISPA
LIS

FORCUTR
LIS