

[illegible]

```

CCCCCCCCC      SSSSSSSSS  PPPPPPPP
CCCCCCCCC      SSSSSSSSS  PPPPPPPP
CC              SS          PP          PP
CC              SS          PP          PP
CC              SS          PP          PP
CC              SS          PP          PP
CC              SSSSSSS    PPPPPPPP
CC              SSSSSSS    PPPPPPPP
CC              SS          PP          PP
CC              SS          PP          PP
CC              SS          PP          PP
CC              SS          PP          PP
CCCCCCCCC      SSSSSSSSS  PP          PP
CCCCCCCCC      SSSSSSSSS  PP

```

```

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
LLLLLLLLLLL

SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
        SSSSSS
        SSSSSS
                                SS
                                SS
                                SS
                                SS
SSSSSSSSS
SSSSSSSSS

```

CS
VO

.....


```

0001 0 - CSP$WAIT does not return status! What if LIB$GET_VM fails?
0002 0
0003 0
0004 0
0005 0 MODULE CSP
0006 0 (IDENT = 'V04-000'
0007 0 ,MAIN = CLUSTER_SERVER
0008 0 ,LANGUAGE (BLISS32)
0009 0 ,ADDRESSING_MODE (EXTERNAL=GENERAL)
0010 0 ) =
0011 1 BEGIN
0012 1 $TITLE 'Cluster Server Process - Main Routine'
0013 1
0014 1 *****
0015 1 *
0016 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0017 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0018 1 * ALL RIGHTS RESERVED.
0019 1 *
0020 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0021 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0022 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0023 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0024 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0025 1 * TRANSFERRED.
0026 1 *
0027 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0028 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0029 1 * CORPORATION.
0030 1 *
0031 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0032 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0033 1 *
0034 1 *
0035 1 *****
0036 1 ++
0037 1
0038 1 FACILITY: Cluster Server Process
0039 1
0040 1 ABSTRACT: Process context for coordinating the actions of cluster
0041 1 servers and event logging in a VAXcluster.
0042 1
0043 1 AUTHOR: Paul R. Beck
0044 1
0045 1 DATE: 03-MAR-1983 Last Edit: 24-AUG-1983 10:32
0046 1
0047 1 REVISION HISTORY:
0048 1
0049 1 V03-010 ADE0003 Alan D. Eldridge 24-Apr-1984
0050 1 Cleaned up some error paths. Use CSP$GL_CURCTX instead
0051 1 of CLX where appropriate.
0052 1
0053 1 V03-009 RSH0125 R. Scott Hanna 22-Mar-1984
0054 1 Remove call to CSP$QUORUM_INIT.
0055 1
0056 1 V03-008 ADE0002 Alan D. Eldridge 28-Feb-1983
0057 1 Change name of CSP$QUORUM to CSP$QUORUM_INIT. Change name

```

```
.. 58      0058 1 | of CSP$$OPCOM to CSP$TELL_OPCOM.
.. 59      0059 1 |
.. 60      0060 1 | V03-007 ADE0001 Alan D. Eldridge 1-Dec-1983
.. 61      0061 1 | Use the ACKMSG service of the connection manager instead
.. 62      0062 1 | of DECnet for inter-node communication.
.. 63      0063 1 |
.. 64      0064 1 | V03-006 PRB0248 Paul Beck 8-Sep-1983
.. 65      0065 1 | Fix problem with the way CSP waits for DECnet availability.
.. 66      0066 1 |
.. 67      0067 1 | V03-005 RSH0060 R. Scott Hanna 24-AUG-1983
.. 68      0068 1 | Add call to CSP$QUORUM during initialization.
.. 69      0069 1 |
.. 70      0070 1 | V03-004 PRB0226 Paul Beck 19-AUG-1983 18:48
.. 71      0071 1 | Start up DECnet object asynchronously from the rest of the
.. 72      0072 1 | initialization, so the scheduler can be running before
.. 73      0073 1 | DECnet has started. Also, remove some excess baggage which
.. 74      0074 1 | would only be needed if DECnet were the permanent mechanism
.. 75      0075 1 | instead of just a holding action.
.. 76      0076 1 |
.. 77      0077 1 | V03-003 PRB0226 Paul Beck 9-JUL-1983 16:39
.. 78      0078 1 | Get requests from CLUB$L_CSPFL, and allow for nonpaged pool
.. 79      0079 1 | (system addresses) therein. Get CSPDEF from LIB$.
.. 80      0080 1 |
.. 81      0081 1 | V03-002 PRB0214 Paul Beck 24-JUN-1983 14:34
.. 82      0082 1 | Change SRC$:CSPDEF to SHRLIB$:CSPDEF
.. 83      0083 1 |
.. 84      0084 1 | V03-001 PRB0200 Paul Beck 6-JUN-1983 21:05
.. 85      0085 1 | Change CTX$ symbols to CLX$ to prevent conflict with RCP.
.. 86      0086 1 | --
```



```

88      0087 1  |
89      0088 1  | External References:
90      0089 1  |
91      0090 1  |
92      0091 1  |
93      0092 1  REQUIRE
94      0093 1  'LIB$:CSPDEF.R32' ;
95      0287 1  LIBRARY
96      0288 1  'SYSS$LIBRARY:LIB' ;
97      0289 1  |
101     0293 1  SWITCHES LIST (SOURCE) ;
102     0294 1  |
103     0295 1  MACRO
104     0296 1  |
105     0297 1  | Define offsets for lock_status.
106     0298 1  |
107     0299 1  LKSB$W_STATUS      = 0,0,16,0 % ;
108     0300 1  LKSB$W_RESERVED    = 2,0,16,0 % ;
109     0301 1  LKSB$L_LKID        = 4,0,32,0 % ;
110     0302 1  LKSB$AB_VALBLK     = 8,0,0,0 % ;
111     0303 1  LKSB$L_VALBLK_KEY  = 8,0,32,0 % ;
112     0304 1  |
113     0305 1  |
114     0306 1  |
115     0307 1  | Linkages used
116     0308 1  |
117     0309 1  LINKAGE
118     0310 1  NO_REGISTERS      = CALL: NOPRESERVE (0,1,2,3,4,5,6,7,8,9,10,11),
119     0311 1  JSB_2             = JSB (REGISTER=2),
120     0312 1  JSB_12            = JSB (REGISTER=1,REGISTER=2),
121     0313 1  JSB_LINKAGE       = JSB,
122     0314 1  LINKDEANONPAGED   = JSB (REGISTER=0): NOPRESERVE(0,1,2,3) ;
123     0315 1  |
124     0316 1  BUILTIN
125     0317 1  INSQUE,
126     0318 1  REMQUE,
127     0319 1  CALLG,
128     0320 1  TESTBITCC,
129     0321 1  TESTBITCS,
130     0322 1  TESTBITSC,
131     0323 1  TESTBITSS ;
132     0324 1  |
```

```
134 0325 1 |
135 0326 1 | External References:
136 0327 1 |
137 0328 1 | EXTERNAL ROUTINE
138 0329 1 | LIB$FREE_VM, | Free virtual memory
139 0330 1 | LIB$GET_VM, | Get virtual memory
140 0331 1 | LIB$GET_EF, | Get event flag
141 0332 1 | EXES$DEANONPAGED : LINKDEANONPAGED, | deallocated nonpaged pool
142 0333 1 | EXES$CSP_COMMAND : JSB_12, | Issue commands to loadable CSP code
143 0334 1 | CSP$SCALL_ACTION : JSB_2, | Action Routine dispatcher
144 0335 1 | CSP$WAIT, | Wait for completion AST
145 0336 1 | CSP$CREATE_CTX : JSB_LINKAGE, | Create new context block
146 0337 1 | CSP$DELETE_CTX : JSB_LINKAGE, | Delete current context block
147 0338 1 | CSP$SAVE_STACK : JSB_LINKAGE, | Save current stack in context block
148 0339 1 | CSP$RESUME : NOVALUE; | Internal completion AST completion
149 0340 1 |
150 0341 1 |
151 0342 1 | Forward References:
152 0343 1 |
153 0344 1 | FORWARD ROUTINE
154 0345 1 | EXIT_HANDLER : NOVALUE, | Exit handler
155 0346 1 | KERNEL_INIT, | kernel mode initialization
156 0347 1 | KERNEL_CLEANUP : NOVALUE, | kernel mode exit handler cleanup
157 0348 1 | SCHEDULE : NO_REGISTERS, | Scheduler for multithreading
158 0349 1 | REMQUE_CSD, | Get local request from CLUB, if any
159 0350 1 | DEANONPAGED, | Kernel routine to call EXES$DEANONPAGED
160 0351 1 | CSP$TELL_OPCODE, | Report error to OPCODE
161 0352 1 | RESUME_THREAD, | Resume execution of thread
162 0353 1 | NEW_REQUEST : JSB_LINKAGE NOVALUE, | Process new client request
163 0354 1 | REPLY, | Tell loadable Exec code to reply
164 0355 1 | KERNEL_ENQW, | $ENQW call from kernel mode
165 0356 1 | CSP$CRASH, | Report bug
166 0357 1 | MUMBLE ;
167 0358 1 |
```



```
169 0359 1  Fixed storage
170 0360 1
171 0361 1
172 0362 1 GLOBAL
173 0363 1   CSP$GL_BASE_FP: LONG,      ! save base FP for scheduler
174 0364 1   CSP$GL_CSPQ: LONG,    ! addr of CLUB$GL_CSPFL queue
175 0365 1   CSP$GQ_RESUME: VECTOR [2,LONG] ! queue of scheduled context blocks
176 0366 1           INITIAL (CSP$GQ_RESUME
177 0367 1           ,CSP$GQ_RESUME
178 0368 1           )
179 0369 1   CSP$GQ_WAIT : VECTOR [2,LONG] ! queue of suspended context blocks
180 0370 1           INITIAL (CSP$GQ_WAIT
181 0371 1           ,CSP$GQ_WAIT
182 0372 1           )
183 0373 1   CSP$GL_CURCTX : REF BLOCK [,BYTE] ; ! Current context block
184 0374 1
185 0375 1 OWN
186 0376 1   CSP$Q_CLX_CSD: VECTOR [2,LONG] ! queue of free CLX/CSD blocks
187 0377 1           INITIAL (CSP$Q_CLX_CSD
188 0378 1           ,CSP$Q_CLX_CSD
189 0379 1           )
190 0380 1   EXIT_REASON : LONG,      ! reason returned to exit handler
191 0381 1   EXIT_HANDLER_BLOCK
192 0382 1           : VECTOR [4,LONG] ! define exit handler
193 0383 1           INITIAL (0,EXIT_HANDLER
194 0384 1           ,1,EXIT_REASON
195 0385 1           )
196 0386 1   LOCK_STATUS : BLOCK [24,BYTE], ! lock status block plus value block
197 0387 1   LOCK_BUFFER : VECTOR [31,BYTE], ! text of lock resource name
198 0388 1   LOCK_NAME : VECTOR [2,LONG] ! working descriptor for lock_buffer
199 0389 1           INITIAL (0,LOCK_BUFFER),
200 0390 1   LOCK_NAME_DESC: VECTOR [2,LONG] ! initial descriptor for lock_buffer
201 0391 1           INITIAL (31,LOCK_BUFFER),
202 0392 1   STARTUP_TIME : VECTOR [2,LONG], ! system time for value block
203 0393 1   BASE_IOSB : VECTOR [2,LONG], ! IOSB for CSP's QIOs
204 0394 1   BASE_EFN : BYTE ; ! allocated event flag for use by CSP
205 0395 1
206 0396 1
207 0397 1  !
208 0398 1  ! Macro to issue call with arguments from kernel mode.
209 0399 1  !
210 M 0400 1 MACRO   KRNL_CALL (K_ROUTINE) =
211 M 0401 1   BEGIN
212 M 0402 1   EXTERNAL ROUTINE SYSS$CMKRNL : ADDRESSING_MODE (ABSOLUTE) ;
213 M 0403 1   BUILTIN SP ;
214 M 0404 1
215 M 0405 1   SYSS$CMKRNL (K_ROUTINE , .SP , %LENGTH - 1
216 M 0406 1   ,%IF %LENGTH GTR 1 %THEN , %REMAINING %FI)
217 0407 1   END% ;
218 0408 1
219 0409 1 MACRO   ELSEIF = ELSE IF % ;
220 0410 1
221 0411 1 ROUTINE FLUSH_LISTING : NOVALUE = ! Force output to .LIS file during
222 0412 1 RETURN ; ! compile
```

.TITLE CSP Cluster Server Process - Main Routine

```
.IDENT \V04-000\
.PSECT $OWNS$,NOEXE,2

00000000' 00000000' 00000 CSP$Q_CLX_CSD:
                                .ADDRESS CSP$Q_CLX_CSD, CSP$Q_CLX_CSD
                                00008 EXIT_REASON:
                                .BLKB 4
00000000 0000C EXIT_HANDLER_BLOCK:
                                .LONG 0
00000000V 00010 .ADDRESS EXIT_HANDLER
00000001 00014 .LONG 1
00000000' 00018 .ADDRESS EXIT_REASON
0001C LOCK_STATUS:
                                .BLKB 24
00034 LOCK_BUFFER:
                                .BLKB 31
00053 .BLKB 1
00000000 00054 LOCK_NAME:
                                .LONG 0
00000000' 00058 .ADDRESS LOCK_BUFFER
0000001F 0005C LOCK_NAME_DESC:
                                .LONG 31
00000000' 00060 .ADDRESS LOCK_BUFFER
00064 STARTUP_TIME:
                                .BLKB 8
0006C BASE_IOSB:
                                .BLKB 8
00074 BASE_EFN:
                                .BLKB 1
.PSECT $GLOBAL$,NOEXE,2

00000 CSP$GL_BASE_FP::
                                .BLKB 4
00004 CSP$GL_CSPQ::
                                .BLKB 4
00000000' 00000000' 00008 CSP$GQ_RESUME::
                                .ADDRESS CSP$GQ_RESUME, CSP$GQ_RESUME
00000000' 00000000' 00010 CSP$GQ_WAIT::
                                .ADDRESS CSP$GQ_WAIT, CSP$GQ_WAIT
00018 CSP$GL_CURCTX::
                                .BLKB 4

.EXTRN LIB$FREE_VM, LIB$GET_VM
.EXTRN LIB$GET_EF, EXE$DEANONPAGED
.EXTRN EXE$CSP_COMMAND
.EXTRN CSP$CALL_ACTION
.EXTRN CSP$WAIT, CSP$CREATE_CTX
.EXTRN CSP$DELETE_CTX
.EXTRN CSP$SAVE_STACK
.EXTRN CSP$RESUME

.PSECT $CODE$,NOWRT,2

0000 00000 FLUSH_LISTING:
                                .WORD Save nothing
                                ; 0411
```


CSP
V04-000

Cluster Server Process - Main Routine

L 9
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 7
(4)

04 00002

RET

; 0412

; Routine Size: 3 bytes, Routine Base: \$CODE\$ + 0000

; 223 0413 1

```
225 0414 1 ROUTINE CLUSTER_SERVER: NOVALUE =
226 0415 2 BEGIN
227 0416 3
228 0417 4 LOCAL
229 0418 5     STATUS,          ! All-purpose completion status
230 0419 6     CLX      : REF BLOCK [,BYTE] ;      ! Ptr to CLX block
231 0420 7
232 0421 8 !
233 0422 9     The maximum number of requests and the maximum CSD size for each
234 0423 0     request are each fixed values. Therefore, allocate and queue one
235 0424 1     CSD per request.
236 0425 2
237 0426 3 INCR I FROM 1 TO CSP$K_MAX_FLWCTL
238 0427 4 DO
239 0428 5     IF (CLX = CSP$CREATE_CTX ()) EQL 0
240 0429 6     THEN
241 0430 7         RETURN (SS$_INSFMEM)
242 0431 8     ELSE
243 0432 9         IF (STATUS = LIB$GET_VM (%REF (CSP$K_MAX_CSDLNG), CLX [CLX$_PO_CSD]))
244 0433 0         THEN
245 0434 1             INSQUE (.CLX, .CSP$Q_CLX_CSD)
246 0435 2         ELSE
247 0436 3             RETURN (.STATUS) ;
248 0437 4
249 0438 5 !
250 0439 6     Perform kernel-mode initialization as needed.
251 0440 7
252 0441 8 IF NOT KRNL_CALL (KERNEL_INIT)
253 0442 9 THEN
254 0443 0     BEGIN
255 0444 1     CSP$TELL_OPCOM
256 0445 2     (%ASCID '%CSP-E-NOCLUSTER, Cluster Server Process exiting: no cluster') ;
257 0446 3     $EXIT (CODE = SS$_ABORT) ;
258 0447 4     END ;
259 0448 5
260 0449 6 !
261 0450 7     Grab an event flag
262 0451 8
263 0452 9 IF NOT (STATUS = LIB$GET_EF (BASE_EFN) ) THEN RETURN .STATUS ;
264 0453 0
265 0454 1 !
266 0455 2     Declare an exit handler to deal with emergencies. In particular, it should
267 0456 3     empty the queue CLUB$GL_CSPFL and restore the blocks to nonpaged pool.
268 0457 4
269 0458 5 IF NOT (STATUS = $DCLEXH (DESBK = EXIT_HANDLER_BLOCK))
270 0459 6 THEN
271 0460 7     RETURN .STATUS ;
272 0461 8
273 0462 9 !
274 0463 0     Set up a condition handler to deal with problems. (Needed?)
275 0464 1
276 0465 2 !MUMBLE ;
277 0466 3
278 0467 4 !
279 0468 5     Request notification of cluster events.
280 0469 6
281 0470 7 !MUMBLE ;
```



```

: 282      0471 2
: 283      0472 2
: 284      0473 2
: 285      0474 2
: 286      0475 2
: 287      0476 2
: 288      0477 2
: 289      0478 2
: 290      0479 1

```

Wait for a request. A request will arrive as an incoming connect request, which is validated, followed by a buffer of data. This data is passed along to the server associated with the connect request.

```

WHILE 1 DO SCHEDULE ();
END ;

```

```

45 54 53 55 4C 43 4F 4E 2D 45 2D 50 53 43 25 00000 P.AAB: .PSECT $PLITS$,NOWRT,NOEXE,2
76 72 65 53 20 72 65 74 73 75 6C 43 20 2C 52 0000F .ASCII \XCSP-E-NOCLUSTER, Cluster Server Process\
6C 63 20 6F 6E 20 3A 67 6E 69 72 65 74 73 75 0001E .ASCII \ exiting: no cluster\
                                010E003C 0003C P.AAA: .LONG 17694780
                                00000000' 00040 .ADDRESS P.AAB

                                .EXTRN SYSS$CMKRNL, SYS$EXIT
                                .EXTRN SYSS$DCLEXH

                                .PSECT $CODE$,NOWRT,2

                                OFFC 00000 CLUSTER_SERVER:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
                                SUBL2 #4, SP
                                MOVL #1, I
                                JSB CSP$CREATE_CTX
                                MOVL R0, CLX
                                BEQL 4$
                                PUSHAB 16(C LX)
                                MOVZWL #4096, 4(SP)
                                PUSHAB 4(SP)
                                CALLS #2, LIB$GET_VM
                                MOVL R0, STATUS
                                BLBC STATUS, 4$
                                INSQUE (CLX), @CSP$Q_CLX_CSD
                                AOBLEQ #8, I, 1$
                                CLRL -(SP)
                                PUSHL SP
                                PUSHAB KERNEL INIT
                                CALLS #3, @SYSS$CMKRNL
                                BLBS R0, 2$
                                PUSHAB P.AAA
                                CALLS #1, CSP$TELL_OPCOM
                                PUSHL #4
                                CALLS #1, SYS$EXIT
                                PUSHAB BASE_EFN
                                CALLS #1, LIB$GET_EF
                                MOVL R0, STATUS
                                BLBC STATUS, 4$
                                PUSHAB EXIT_HANDLER_BLOCK
                                CALLS #1, SYSS$DCLEXH

```

0414
0426
0428
0432
0434
0428
0441
0445
0446
0452
0458

0000V	54 07 CF	50 54 00 F9	D0 00075 E9 00078 FB 0007B 11 00080 04 00082	3\$: 4\$:	MOVL BLBC CALLS BRB RET	RO, STATUS STATUS, 4\$ #0, SCHEDULE 3\$
-------	----------------	----------------------	--	--------------	-------------------------------------	--

0478
0479

```
; Routine Size: 131 bytes,    Routine Base: $CODE$ + 0003
```

: 291 0480 1


```

: 293 0481 1 %SBTTL 'SCHEDULE - schedule new requests, resume suspended threads'
: 294 0482 1 ROUTINE SCHEDULE: NO_REGISTERS =
: 295 0483 1
: 296 0484 1 ++
: 297 0485 1
: 298 0486 1 SCHEDULE is the thread scheduler. It hibernates when there is nothing to do.
: 299 0487 1 When wakened, it removes items from one of the following queues:
: 300 0488 1
: 301 0489 1 1. New Requests - containing buffers which are sent off to the
: 302 0490 1 appropriate servers.
: 303 0491 1 2. Thread Resumptions - containing context blocks to be resumed.
: 304 0492 1
: 305 0493 1 It continues to service these queues until both are empty, then
: 306 0494 1 hibernates once more.
: 307 0495 1
: 308 0496 1 CALLING SEQUENCE:
: 309 0497 1 CALL
: 310 0498 1
: 311 0499 1 FORMAL PARAMETERS:
: 312 0500 1 None.
: 313 0501 1
: 314 0502 1 COMPLETION CODES:
: 315 0503 1 None. SCHEDULE runs ad nauseum.
: 316 0504 1 --
: 317 0505 2 BEGIN
: 318 0506 2 LOCAL
: 319 0507 2 CSD : REF BLOCK [,BYTE], ! new local request
: 320 0508 2 STATUS ;
: 321 0509 2
: 322 0510 2 BUILTIN FP ;
: 323 0511 2
: 324 0512 2 Save the frame pointer, enabling the scheduler to be reentered from
: 325 0513 2 the context save routines.
: 326 0514 2
: 327 0515 2 CSP$GL_BASE_FP = .FP ;
: 328 0516 2
: 329 0517 2
: 330 0518 2 Try for something to do, and hibernate if there's nothing.
: 331 0519 2 If a thread is active (CSP$GL_CURCTX non-zero) then there's a bug.
: 332 0520 2
: 333 0521 2 WHILE .CSP$GL_CURCTX EQL 0
: 334 0522 2 DO
: 335 0523 2 IF NOT REMQUE (.CSP$GQ_RESUME, CSP$GL_CURCTX) THEN
: 336 0524 2
: 337 0525 2 Resume a suspended thread. Context block has been placed in the
: 338 0526 2 grant queue by an AST.
: 339 0527 2
: 340 0528 2 ** Note that RESUME_THREAD does not return in-line **
: 341 0529 2 ** the scheduler will be reentered from the top **
: 342 0530 2
: 343 0531 2 IF TESTBITCC (CSP$GL_CURCTX [CLX$V_QUEUED])
: 344 0532 2 THEN CSP$$CRASH (SS$-NOPRIVSTRT + T6)
: 345 0533 2 ELSE RESUME_THREAD (J
: 346 0534 2
: 347 0535 2 ELSE
: 348 0536 2 If we have a free process space CSD then service a new request
: 349 0537 2
```


CSP
V04-000

Cluster Server Process - Main Routine
SCHEDULE - schedule new requests, resume suspen

D 10
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1

Page 12
(6)

```
: 350      0538 2      IF KRNL CALL (REMQUE_CSD) THEN
: 351      0539 2          NEW_REQUEST ( )
: 352      0540 2      ELSE
: 353      0541 2          $HIBER ;
: 354      0542 2
: 355      0543 2 CSP$$CRASH (SS$_BADPARAM);
: 356      0544 2 RETURN 0 ;
: 357      0545 1 END ;
```

.EXTRN SYSSHIBER

0000 00000 SCHEDULE:				
0000'	CF	5D D0 00002	1\$: .WORD Save nothing	: 0482
		0000' CF D5 00007	MOVL FP, CSP\$GL_BASE_FP	: 0515
		46 12 0000B	TSTL CSP\$GL_CURCTX	: 0521
0000'	CF	0000' DF 0F 0000D	BNEQ 5\$	
		1D 1D 00014	REMQUE @CSP\$GQ_RESUME, CSP\$GL_CURCTX	: 0523
	50	0000' CF D0 00016	BVS 3\$	
OC	OB	00 00 0001B	MOVL CSP\$GL_CURCTX, R0	: 0531
	7E	2810 8F 3C 00020	BBSC #0, 11(R0), 2\$	
0000V	CF	01 FB 00025	MOVZWL #10256, -(SP)	: 0532
		DB 11 0002A	CALLS #1, CSP\$\$CRASH	
0000V	CF	00 FB 0002C	BRB 1\$	
		D4 11 00031	CALLS #0, RESUME_THREAD	: 0533
		7E D4 00033	BRB 1\$	: 0531
		5E DD 00035	CLRL -(SP)	: 0538
		0000V CF 9F 00037	PUSHL SP	
00000000G	9F	03 FB 0003B	PUSHAB REMQUE_CSD	
	05	50 E9 00042	CALLS #3, @SYSCMKRNL	
		0000V 30 00045	BLBC R0, 4\$	
		BD 11 00048	BSBW NEW_REQUEST	: 0539
00000000G	00	00 FB 0004A	BRB 1\$	
		B4 11 00051	CALLS #0, SYSSHIBER	: 0540
		14 DD 00053	BRB 1\$	: 0523
0000V	CF	01 FB 00055	PUSHL #20	: 0543
		50 D4 0005A	CALLS #1, CSP\$\$CRASH	
		04 0005C	CLRL R0	: 0544
			RET	: 0545

; Routine Size: 93 bytes, Routine Base: \$CODE\$ + 0086

; 358 0546 1


```

360 0547 1 %SBTTL 'NEW_REQUEST - process a new request'
361 0548 1 ROUTINE NEW_REQUEST : JSB_LINKAGE NOVALUE =
362 0549 1 ++
363 0550 1
364 0551 1 Dispatch a new execution thread .
365 0552 1
366 0553 1
367 0554 1 CALLING SEQUENCE:
368 0555 1 JSB
369 0556 1
370 0557 1 FORMAL PARAMETERS:
371 0558 1 None
372 0559 1
373 0560 1 --
374 0561 2 BEGIN
375 0562 2 LOCAL
376 0563 2 CSD : REF BLOCK [,BYTE], ! Context block
377 0564 2 STATUS ;
378 0565 2
379 0566 2 CSD = .CSP$GL_CURCTX [CLX$A_PO_CSD] ; ! Get the P0 space CSD
380 0567 2
381 0568 2
382 0569 2 Relocate pointers in CSD, dispatch to action routine for this client,
383 0570 2 respond to EXE$CSP_COMMAND when done.
384 0571 2
385 0572 2 CSD [CSD$L_SENDOFF] = .CSD [CSD$L_SENDOFF] + .CSD ;
386 0573 2 CSD [CSD$L_RECVOFF] = .CSD [CSD$L_RECVOFF] + .CSD ;
387 0574 2
388 0575 2 STATUS = CSP$CALL_ACTION (.CSD) ;
389 0576 2
390 0577 2 KRNL_CALL (REPLY) ;
391 0578 2
392 0579 2 INSQUE (.CSP$GL_CURCTX, CSP$Q_CLX_CSD) ;
393 0580 2 CSP$GL_CURCTX = 0 ;
394 0581 2
395 0582 2 RETURN
396 0583 1 END ;
```

			52	DD	00000	NEW_REQUEST:			
						PUSHL	R2		0548
		50	CF	D0	00002	MOVL	CSP\$GL_CURCTX, R0		0566
		52	A0	D0	00007	MOVL	16(R0), CSD		
16	A2		52	C0	0000B	ADDL2	CSD, 22(CSD)		0572
1E	A2		52	C0	0000F	ADDL2	CSD, 30(CSD)		0573
		00000000G	00	16	00013	JSB	CSP\$CALL_ACTION		0575
			7E	D4	00019	CLRL	-(SP)		0577
			5E	DD	0001B	PUSHL	SP		
		0000V	CF	9F	0001D	PUSHAB	REPLY		
00000000G	9F		03	FB	00021	CALLS	#3, @#SYSS\$CMKRNL		
0000'	CF	0000'	DF	0E	00028	INSQUE	@CSP\$GL_CURCTX, CSP\$Q_CLX_CSD		0579
		0000'	CF	D4	0002F	CLRL	CSP\$GL_CURCTX		0580
			04	BA	00033	POPR	#^M<R25		0583
			05	00	0035	RSB			

CSP
V04-000

Cluster Server Process - Main Routine
NEW_REQUEST - process a new request

F 10
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 14
(7)

; Routine Size: 54 bytes, Routine Base: \$CODE\$ + 00E3

; 397 0584 1

CSP
V04-000

Cluster Server Process - Main Routine

REPLY - Call loadable Exec code to finish block

G 10

16-Sep-1984 01:13:39

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 15

(8)

```

399 0585 1 XSBTTL 'REPLY - Call loadable Exec code to finish block transfer'
400 0586 1 ROUTINE REPLY =
401 0587 1 ++
402 0588 1
403 0589 1 Client is done. Copy the response (if any) and give then S0 space CSD back
404 0590 1 to the loadable Exec code.
405 0591 1
406 0592 1
407 0593 1 CALLING SEQUENCE:
408 0594 1 CALL in kernel mode at IPL 0
409 0595 1
410 0596 1 FORMAL PARAMETERS:
411 0597 1 None
412 0598 1
413 0599 1 --
414 0600 2 BEGIN
415 0601 2 BIND SO_CSD = .CSP$GL_CURCTX [CLX$A_SO_CSD] : BLOCK [,BYTE],
416 0602 2 PO_CSD = .CSP$GL_CURCTX [CLX$A_PO_CSD] : BLOCK [,BYTE];
417 0603 2
418 0604 2 LOCAL SIZE, COMMAND ;
419 0605 2
420 0606 2 SIZE = .PO_CSD [CSD$L_RECVLEN] ;
421 0607 2
422 0608 2
423 0609 2
424 0610 2 SO_CSD [CSD$L_RECVOFF] contains a real offset, but PO_CSD [CSD$L_RECVOFF]
425 0611 2 has been converted to a pointer.
426 0612 2
427 0613 2 Determine the proper response code and move whatever data needs moving.
428 0614 2
429 0615 2
430 0616 2 IF (.SIZE + .SO_CSD [CSD$L_RECVOFF]) GTRU .SO_CSD [CSD$W_SIZE] THEN
431 0617 2 BEGIN
432 0618 2 SIZE = .SO_CSD [CSD$W_SIZE] - .SO_CSD [CSD$L_RECVOFF] ;
433 0619 2 COMMAND = CSP$_BAD_CSD ;
434 0620 2 END
435 0621 2
436 0622 2 ELSEIF .PO_CSD [CSD$L_RECVLEN] NEQ 0 THEN
437 0623 2 COMMAND = CSP$_REPLY
438 0624 2 ELSE
439 0625 2 COMMAND = CSP$_DONE ;
440 0626 2
441 0627 2 CH$MOVE (.SIZE
442 0628 2 , .PO_CSD [CSD$L_RECVOFF] ! Already a pointer
443 0629 2 , .SO_CSD [CSD$L_RECVOFF] + .SO_CSD ! Calculate pointer
444 0630 2 ;
445 0631 2
446 0632 2
447 0633 2 Pass the CSD back to the loadable Exec CSD code and erase the CSD pointer
448 0634 2
449 0635 2
450 0636 2 EXEC$CSP_COMMAND (.COMMAND, .CSP$GL_CURCTX [CLX$A_SO_CSD]) ;
451 0637 2
452 0638 2 CSP$GL_CURCTX [CLX$A_SO_CSD] = 0 ;
453 0639 2
454 0640 2 RETURN 1 ;
455 0641 1 END ;
```

53	08	53 A1	56	0000'	CF	D0	00002	REPLY:	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	:	0586
			51	0C	A6	7D	00007		MOVL	CSP\$GL_CURCTX, R6	:	0601
			50	1A	A2	D0	0000B		MOVQ	12(R6), R1	:	
			50	1E	A1	C1	0000F		MOVL	26(R2), SIZE	:	0606
			10		00	ED	00014		ADDL3	30(R1), SIZE, R3	:	0616
					0D	1E	0001A		CMPZV	#0, #16, 8(R1), R3	:	
			50	08	A1	3C	0001C		BGEQU	1\$	:	
			50	1E	A1	C2	00020		MOVZWL	8(R1), SIZE	:	0618
			57		03	D0	00024		SUBL2	30(R1), SIZE	:	
					0D	11	00027		MOVL	#3, COMMAND	:	0619
				1A	A2	D5	00029	1\$:	BRB	3\$	:	0616
					05	13	0002C		TSTL	26(R2)	:	0622
			57		05	D0	0002E		BEQL	2\$	:	
					03	11	00031		MOVL	#5, COMMAND	:	0623
			57		04	D0	00033	2\$:	BRB	3\$	:	
		51	57		61	C1	00036	3\$:	MOVL	#4, COMMAND	:	0625
		61	A1		50	28	0003B		ADDL3	(R1), 30(R1), R1	:	0629
			B2		57	D0	00044		MOVC3	SIZE, @30(R2), (R1)	:	
			52	0C	A6	D0	00040		MOVL	12(R6), R2	:	0636
			51		57	D0	00047		MOVL	COMMAND, R1	:	
				00000000G	00	16	00047		JSB	EXE\$CSP, COMMAND	:	
			50	0000'	CF	D0	0004D		MOVL	CSP\$GL_CURCTX, R0	:	0638
				0C	A0	D4	00052		CLRL	12(R0)	:	
			50		01	D0	00055		MOVL	#1, R0	:	0640
					04	00058			RET		:	0641

; Routine Size: 89 bytes, Routine Base: \$CODE\$ + 0119

; 456 0642 1


```

458 0643 1 %SBTTL 'RESUME_THREAD - resume execution of suspended thread'
459 0644 1 ROUTINE RESUME_THREAD =
460 0645 1
461 0646 1 |++
462 0647 1 | Restore the context of a thread and resume its execution. The context
463 0648 1 | consists of the stack and registers. The top of the saved stack contains a
464 0649 1 | call frame which points to the resume PC and contains the thread's registers.
465 0650 1 | So by restoring the stack and "return"ing, we resume the thread.
466 0651 1 |
467 0652 1 | CALLING SEQUENCE:
468 0653 1 | CALL (from SCHEDULE only!)
469 0654 1 |
470 0655 1 | FORMAL PARAMETERS:
471 0656 1 | None
472 0657 1 |
473 0658 1 | COMPLETION CODES:
474 0659 1 | NA, R0 is restored from its saved state.
475 0660 1 |
476 0661 1 |--
477 0662 2 BEGIN
478 0663 2
479 0664 2 REGISTER
480 0665 2 | SAVE_R0,SAVE_R1, | temp save registers R0,R1
481 0666 2 | STATUS ; | completion status
482 0667 2
483 0668 2 BUILTIN
484 0669 2 | R0,R1,FP,SP ;
485 0670 2
486 0671 2 |
487 0672 2 | Thread resumption consists of simply restoring the exact stack as it existed
488 0673 2 | for the suspended thread, by copying it from the context block. Then kill the
489 0674 2 | context block and restore R0,R1. The other registers are restored by the
490 0675 2 | RETURN, which is logically a RETURN from CSP$WAIT.
491 0676 2 |
492 0677 2 |
493 0678 2
494 0679 2 | SAVE_R0 = .CSP$GL_CURCTX [CLX$R0] ;
495 0680 2 | SAVE_R1 = .CSP$GL_CURCTX [CLX$R1] ;
496 0681 2
497 0682 2 | SP = .CSP$GL_BASE_FP - .CSP$GL_CURCTX [CLX$STACKSIZE] ;
498 0683 2 | FP = .SP ;
499 0684 2
500 0685 2 | CH$MOVE (.CSP$GL_CURCTX [CLX$STACKSIZE], .CSP$GL_CURCTX [CLX$STACK], .FP) ;
501 0686 2
502 0687 2 |
503 0688 2 | Deallocate the saved stack, then restore the saved context by
504 0689 2 | simply returning to it.
505 0690 2
506 0691 2 | IF TESTBITCC (CSP$GL_CURCTX [CLX$LOCAL_STACK]) THEN
507 0692 2 | LIB$FREE_VM (CSP$GL_CURCTX [CLX$STACKSIZE], CSP$GL_CURCTX [CLX$STACK]) ;
508 0693 2
509 0694 2 | R0 = .SAVE_R0 ;
510 0695 2 | R1 = .SAVE_R1 ;
511 0696 2
512 0697 2 | RETURN (.R0) ; | This restores reg's and resumes thread
513 0698 1 END ;
```

```
01FC 00000 RESUME_THREAD:
      58 0000' CF D0 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8      : 0644
      56 28 A8 7D 00007      MOVL      CSP$GL_CURCTX, R8                : 0679
      38 38 A8 C3 0000B      MOVQ      40(R8), SAVE_R0                    :
      5D 5E D0 00012      SUBL3      56(R8), CSP$GL_BASE_FP, SP          : 0682
      38 A8 28 00015      MOVL      SP, FP                                : 0683
      03 E4 0001B      MOVQ      56(R8), @60(R8), (FP)                    : 0685
      3C A8 9F 00020      BBSC      #3, 11(R8), 1$                        : 0691
      38 A8 9F 00023      PUSHAB     60(R8)                                : 0692
      02 FB 00026      PUSHAB     56(R8)
      56 7D 0002D 1$:      CALLS     #2, LIB$FREE_VM
      04 00030      MOVQ      SAVE_R0, R0
      RET
```

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 0172

; 514 0699 1

CSP
V04-000

Cluster Server Process - Main Routine
RESUME_THREAD - resume execution of suspended t

K 10
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 19
(10)

```
: 516 0700 1
: 517 0701 1 %SBTTL 'KERNEL_ENQW - $ENQW call from kernel mode'
: 518 0702 1
: 519 0703 1 GLOBAL ROUTINE KERNEL_ENQW =
: 520 0704 1
: 521 0705 1 !++
: 522 0706 1
: 523 0707 1 Issues $ENQW call from kernel mode.
: 524 0708 1
: 525 0709 1 CALLING SEQUENCE:
: 526 0710 1 CALL
: 527 0711 1
: 528 0712 1 FORMAL PARAMETERS:
: 529 0713 1 Call parameters identical to those for $ENQW;
: 530 0714 1 the argument list is simply passed on.
: 531 0715 1
: 532 0716 1 COMPLETION CODES:
: 533 0717 1 As from $ENQW, plus SS$_NOPRIV if process lacks CMKRNL.
: 534 0718 1
: 535 0719 1 !--
: 536 0720 2 BEGIN
: 537 0721 2 EXTERNAL ROUTINE SYS$ENQW ;
: 538 0722 2 BUILTIN AP ;
: 539 0723 2
: 540 0724 2 !RETURN $CMKRNL ( ROUTIN = SYS$ENQW, ARGLST = .AP ) ;
: 541 0725 2 RETURN CALLG (.AP, SYS$ENQW) ;
: 542 0726 1 END ;
```

.EXTRN SYS\$ENQW

00000000G 00 0000 00000
6C FA 00002
04 00009

.ENTRY KERNEL_ENQW, Save nothing
CALLG (AP), SYS\$ENQW
RET

: 0703
: 0725
: 0726

: Routine Size: 10 bytes, Routine Base: \$CODE\$ + 01A3

: 543 0727 1

```
.PSECT $CODE$,NOWRT,2
```


CSP
V04-000

Cluster Server Process - Main Routine
CSP\$TELL_OPCOM - operator communications

M 10
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 21
(11)

					007C	00000
				AC	D0	00002
0000'	CF		56	66	28	00006
0000'	CF	04	B6	08	C1	0000D
			66	7E	D4	00013
				CF	9F	00015
		00000000G	00	02	FB	00019
				04	00	00020

.ENTRY	CSP\$TELL_OPCOM, Save R2,R3,R4,R5,R6	: 0730
MOVL	MESSAGE, R6	: 0757
MOVC3	(R6), @4(R6), OP_BUF+8	:
ADDL3	#8, (R6), OP_DESC	: 0762
CLRL	-(SP)	: 0767
PUSHAB	OP_DESC	:
CALLS	#2, SYS\$SNDOPR	:
RET		: 0768

; Routine Size: 33 bytes, Routine Base: \$CODE\$ + 01AD

; 586 0769 1

CSP
V04-000

Cluster Server Process - Main Routine
EXIT_HANDLER

N 10
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 22
(12)

```

: 588 0770 1 %SBTTL 'EXIT_HANDLER'
: 589 0771 1 ROUTINE EXIT_HANDLER : NOVALUE =
: 590 0772 1
: 591 0773 1 ++
: 592 0774 1
: 593 0775 1 Image exit handler; return excess items in CLUB$GL_CSPFL to nonpaged pool.
: 594 0776 1 Also report the problem.
: 595 0777 1
: 596 0778 1 CALLING SEQUENCE:
: 597 0779 1
: 598 0780 1 FORMAL PARAMETERS:
: 599 0781 1
: 600 0782 1 COMPLETION CODES:
: 601 0783 1
: 602 0784 1 --
: 603 0785 2 BEGIN
: 604 0786 2 LOCAL
: 605 0787 2 CSD ;
: 606 0788 2
: 607 0789 2 CSP$TELL OPCOM ( %ASCID 'CSP-E-EXIT, Cluster Server Process exiting' ) ;
: 608 0790 2 KRNL_CALL ( KERNEL_CLEANUP ) ;
: 609 0791 2
: 610 0792 2 RETURN
: 611 0793 1 END ;
```

```

75 6C 43 20 2C 54 49 58 45 2D 45 2D 50 53 43 00044 P.AAD: .PSECT $SPLITS,NOWRT,NOEXE,2
6F 72 50 20 72 65 76 72 65 53 20 72 65 74 73 00053 .ASCII \CSP-E-EXIT, Cluster Server Process exiti\
: 69 74 69 78 65 20 73 73 65 63 00062
: 00 00 67 6E 0006C
: 010E002A 00070 P.AAC: .ASCII \ng\<0><0>
: 00000000 00074 .LONG 17694762
: .ADDRESS P.AAD
```

```

: .PSECT $CODE$,NOWRT,2
: 0000 0000 EXIT_HANDLER:
: .WORD Save nothing : 0771
: PUSHAB P.AAC : 0789
: CALLS #1, CSP$TELL_OPCOM :
: CLRL -(SP) : 0790
: PUSHL SP :
: PUSHAB KERNEL_CLEANUP :
: CALLS #3, @#SYS$CMKRNL :
: RET : 0793
```

; Routine Size: 26 bytes, Routine Base: \$CODE\$ + 01CE

; 612 0794 1


```

: 614 0795 1 %SBTTL 'KERNEL_INIT'
: 615 0796 1 ROUTINE KERNEL_INIT =
: 616 0797 1
: 617 0798 1 !++
: 618 0799 1
: 619 0800 1 Perform kernel mode initialization:
: 620 0801 1
: 621 0802 1 Initialize queue in CLUB structure, fill in CLUB$GL_CSPIPID.
: 622 0803 1
: 623 0804 1 CALLING SEQUENCE:
: 624 0805 1 $CMKRNL ( KERNEL_INIT )
: 625 0806 1
: 626 0807 1 FORMAL PARAMETERS:
: 627 0808 1 none
: 628 0809 1
: 629 0810 1 COMPLETION CODES:
: 630 0811 1
: 631 0812 1 --
: 632 0813 2 BEGIN
: 633 0814 2 EXTERNAL
: 634 0815 2 CLUB$GL_CLUB : REF BLOCK [,BYTE], ! cluster block
: 635 0816 2 SCH$GL_CURPCB
: 636 0817 2 : REF BLOCK [,BYTE] ; ! current PCB.
: 637 0818 2
: 638 0819 2 IF CLUB$GL_CLUB NEQ 0 THEN
: 639 0820 2 BEGIN
: 640 0821 2
: 641 0822 2 Init the queue
: 642 0823 2
: 643 0824 2 CSP$GL_CSPQ =
: 644 0825 2 CLUB$GL_CLUB [CLUB$GL_CSPFL] =
: 645 0826 2 CLUB$GL_CLUB [CLUB$GL_CSPBL] =
: 646 0827 2 CLUB$GL_CLUB [CLUB$GL_CSPFL] ;
: 647 0828 2
: 648 0829 2 CLUB$GL_CLUB [CLUB$GL_CSPIPID] = .SCH$GL_CURPCB [PCB$GL_PID] ;
: 649 0830 2
: 650 0831 2 RETURN 1 ;
: 651 0832 2 END
: 652 0833 2 ELSE
: 653 0834 2
: 654 0835 2 Not in the cluster
: 655 0836 2
: 656 0837 2 RETURN 0 ;
: 657 0838 1 END ;
```

.EXTRN CLUB\$GL_CLUB, SCH\$GL_CURPCB

0000 00000 KERNEL_INIT:						
				.WORD	Save nothing	: 0796
				MOVL	CLUB\$GL_CLUB, R1	: 0819
				BEQL	1\$	:
				MOVAB	136(R1), R0	: 0827
008C	C1			MOVL	R0, 140(R1)	:
0088	C1			MOVL	R0, 136(R1)	: 0826
0000	CF			MOVL	R0, CSP\$GL_CSPQ	: 0825

CSP
V04-000

Cluster Server Process - Main Routine
KERNEL_INIT

C 11
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1

Page 24
(13)

0090	50	00000000G	00	D0	0001F	MOVL	SCH\$GL_CURPCB, R0	:	0829
	C1	60	A0	D0	00026	MOVL	96(R0); 144(R1)	:	
	50		01	D0	0002C	MOVL	#1, R0	:	0837
				04	0002F	RET		:	
			50	D4	00030	CLRL	R0	:	
			04	00032	1\$:	RET		:	0838

; Routine Size: 51 bytes, Routine Base: \$CODE\$ + 01E8

; 658 0839 1

CSP
V04-000

Cluster Server Process - Main Routine
KERNEL_CLEANUP

D 11
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 25
(14)

```
: 660      0840 1 %SBTTL 'KERNEL_CLEANUP'
: 661      0841 1 ROUTINE KERNEL_CLEANUP : NOVALUE =
: 662      0842 1
: 663      0843 1 !++
: 664      0844 1
: 665      0845 1 Perform kernel-mode exit handler stuff: reset the CLUB so that
: 666      0846 1 it's clear the CSP has disappeared.
: 667      0847 1
: 668      0848 1 CALLING SEQUENCE:
: 669      0849 1
: 670      0850 1 FORMAL PARAMETERS:
: 671      0851 1
: 672      0852 1 COMPLETION CODES:
: 673      0853 1
: 674      0854 1 !--
: 675      0855 2 BEGIN
: 676      0856 2 EXTERNAL
: 677      0857 2 CLUSGL_CLUB : REF BLOCK [,BYTE] ;
: 678      0858 2
: 679      0859 2 LOCAL
: 680      0860 2 CSD ;
: 681      0861 2
: 682      0862 2 CLUSGL CLUB [CLUB$L CSPIPID] = 0 ;
: 683      0863 2 UNTIL REMQUE (.CLUSGL_CLUB [CLUB$L_CSPFL], CSD)
: 684      0864 2 DO
: 685      0865 2 EXE$CSP_COMMAND (CSP$_ABORT, .CSD) ;
: 686      0866 2
: 687      0867 2 RETURN ;
: 688      0868 1 END ;
```

OFFC 00000 KERNEL_CLEANUP:

53	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 0841
50		63	D0	00009	MOVAB	CLUSGL_CLUB, R3	: 0862
	0090	C0	D4	0000C	MOVL	CLUSGL_CLUB, R0	: 0863
50		63	D0	00010	CLRL	144(R0)	: 0865
52	0088	D0	0F	00013	MOVL	CLUSGL_CLUB, R0	: 0868
		0B	1D	00018	REMQUE	@136(R0), CSD	
51		02	D0	0001A	BVS	2\$	
	00000000G	00	16	0001D	MOVL	#2, R1	
		EB	11	00023	JSB	EXE\$CSP_COMMAND	
			04	00025	BRB	1\$	
					RET		

: Routine Size: 38 bytes, Routine Base: \$CODE\$ + 021B

: 689 0869 1

```

: 691 0870 1 %SBTTL 'REMQUE_CSD'
: 692 0871 1 ROUTINE REMQUE_CSD =
: 693 0872 1
: 694 0873 1 !++
: 695 0874 1
: 696 0875 1 Go to kernel mode and read an entry from the queue CLUB$GL_CSPFL in the
: 697 0876 1 system CLUB structure. If we get one, copy it to the CSD structure and
: 698 0877 1 deallocate the nonpaged pool.
: 699 0878 1
: 700 0879 1 CALLING SEQUENCE:
: 701 0880 1 STATUS = REMQUE_CSD ()
: 702 0881 1
: 703 0882 1 FORMAL PARAMETERS:
: 704 0883 1 None
: 705 0884 1
: 706 0885 1 IMPLICIT OUTPUTS:
: 707 0886 1 CSP$GL_CURCTX Address of CLX block if work was found
: 708 0887 1 0 if there's nothing to do
: 709 0888 1
: 710 0889 1 COMPLETION CODES:
: 711 0890 1 0 = queue was empty (should not happen).
: 712 0891 1 1 = got an entry
: 713 0892 1
: 714 0893 1 --
: 715 0894 2 BEGIN
: 716 0895 2 LOCAL QUIT : INITIAL (0),
: 717 0896 2 SO_CSD : REF BLOCK [,BYTE] ; ! Nonpaged pool CSD ptr
: 718 0897 2
: 719 0898 2 IF ..CSP$GL_CSPQ EQL 0 THEN RETURN 0 ; ! Not in cluster yet
: 720 0899 2
: 721 0900 2 IF NOT REMQUE (.CSP$Q_CLX_CSD, CSP$GL_CURCTX)
: 722 0901 3 THEN BEGIN
: 723 0902 3 UNTIL .QUIT
: 724 0903 3 DO IF REMQUE (..CSP$GL_CSPQ, SO_CSD)
: 725 0904 3 THEN QUIT = 1
: 726 0905 3 ELSE IF .SO_CSD [CSD$B_TYPE] NEQ DYN$C CLU
: 727 0906 3 OR .SO_CSD [CSD$W_SIZE] GTRU CSP$R_MAX_CSDLNG
: 728 0907 3 THEN EXE$CSP_COMMAND (CSP$BAD_CSD, .SO_CSD)
: 729 0908 4 ELSE BEGIN
: 730 0909 4 CSP$GL_CURCTX [CLX$A_SO_CSD] = .SO_CSD ;
: 731 0910 4 CSP$GL_CURCTX [CLX$B_FLAGS] = 0 ;
: 732 0911 4 CH$MOVE (.SO_CSD [CSD$W_SIZE]
: 733 0912 4 ..SO_CSD
: 734 0913 4 ;.CSP$GL_CURCTX [CLX$A_PO_CSD]
: 735 0914 4 ) ;
: 736 0915 4 RETURN 1 ;
: 737 0916 3 END ;
: 738 0917 3
: 739 0918 3 INSQUE (.CSP$GL_CURCTX, CSP$Q_CLX_CSD) ;
: 740 0919 2 END ;
: 741 0920 2
: 742 0921 2 CSP$GL_CURCTX = 0 ;
: 743 0922 2 RETURN 0 ;
: 744 0923 1 END ;
```



```
OFFC 00000 REMQUE_CSD:
      58      0000' CF 9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      : 0871
      EC      B8 D5 00009      MOVAB      CSP$GL_CURCTX, R8      : 0894
      52      13 0000C      CLRL      QUIT      : 0898
      68      0000' DF 0F 0000E      TSTL      @CSP$GL_CSPQ      : 0900
      49      1D 00013      BEQL      7$      : 0902
      40      57 E8 00015 1$:      BLBS      QUIT, 5$      : 0903
      50      EC B8 9E 00018      MOVAB      @CSP$GL_CSPQ, R0      : 0904
      56      00 B0 0F 0001C      REMQUE      @0(R0), -SO_CSD      : 0905
      57      05 1C 00020      BVC      2$      : 0906
      65      8F      0A A6 91 00027 2$:      CMPL      #1, QUIT      : 0907
      1000    8F      08 A6 B1 0002E      BRB      1$      : 0909
      52      0E 1B 00034      CMPB      10(SO_CSD), #101      : 0910
      51      03 D0 00039      BNEQ      3$      : 0913
      00000000G 00 16 0003C      CMPW      8(SO_CSD), #4096      : 0915
      50      D1 11 00042      BLEQU      4$      : 0918
      0C      50      68 D0 00044 4$:      MOVL      SO_CSD, R2      : 0921
      A0      56 D0 00047      MOVL      #3, R1      : 0923
      10      B0      0B A0 94 0004B      MOVL      SO_CSD, 12(R0)      :
      66      08 A6 28 0004E      CLRB      11(R0)      :
      50      01 D0 00054      MOVC3      8(SO_CSD), (SO_CSD), @16(R0)      :
      0000' CF      00 B8 0E 00058 5$:      MOVL      #1, R0      :
      68      D4 0005E 6$:      RET      :
      50      D4 00060 7$:      INSQUE      @CSP$GL_CURCTX, CSP$Q_CLX_CSD      :
      04      04 00062      CLRL      CSP$GL_CURCTX      :
      RET      R0      : 0923
```

; Routine Size: 99 bytes, Routine Base: \$CODE\$ + 0241

; 745 0924 1

CSP
V04-000

Cluster Server Process - Main Routine
DEANONPAGED - call EXES\$DEANONPAGED from kernel

G 11
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 28
(16)

```
: 747      0925 1 %SBTTL 'DEANONPAGED - call EXES$DEANONPAGED from kernel mode'
: 748      0926 1 ROUTINE DEANONPAGED (BUFFER) =
: 749      0927 1
: 750      0928 1 ++
: 751      0929 1
: 752      0930 1 Call EXES$DEANONPAGED from kernel mode
: 753      0931 1
: 754      0932 1 CALLING SEQUENCE:
: 755      0933 1 KRNL_CALL (DEANONPAGED, BUF) in Kernel mode
: 756      0934 1
: 757      0935 1 FORMAL PARAMETERS:
: 758      0936 1 BUF = address of buffer to deallocate (SIZE field indicates how big)
: 759      0937 1
: 760      0938 1 COMPLETION CODES:
: 761      0939 1
: 762      0940 1 --
: 763      0941 1 RETURN EXES$DEANONPAGED ( .BUFFER ) ;
```

```
OFFC 00000 DEANONPAGED:
50      04 AC D0 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
00000000G 00 16 00006 MOVL BUFFER, R0
04 0000C 04 0000C JSB EXES$DEANONPAGED
RET
```

```
: 0926
: 0941
:
```

; Routine Size: 13 bytes, Routine Base: \$CODE\$ + 02A4

; 764 0942 1

CSP
V04-000

Cluster Server Process - Main Routine
MUMBLE - Dummy routine

H 11
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 29
(17)

: 766
: 767
: 768
0943 1 %SBTTL 'MUMBLE - Dummy routine'
0944 1
0945 1 GLOBAL ROUTINE MUMBLE = RETURN 1 ;

50 0000 00000
01 D0 00002
04 00005

.ENTRY MUMBLE, Save nothing
MOVL #1, R0
RET

: 0945
:
:

; Routine Size: 6 bytes, Routine Base: \$CODE\$ + 02B1

: 769
: 770
: 771
: 772
: 773
: 774
: 775
: 776
: 777
: 778
: 779
: 780
0946 1
0947 1
0948 1
0949 1
0950 1 %SBTTL 'CSP\$\$CRASH - Report bug'
0951 1
0952 1 GLOBAL ROUTINE CSP\$\$CRASH (CODE) =
0953 1
0954 2 BEGIN
0955 2 \$EXIT (CODE = .CODE) ;
0956 2 RETURN .CODE ;
0957 1 END ;

00000000G 00 04 AC DD 00002
50 04 AC D0 0000C
04 00010

.ENTRY CSP\$\$CRASH, Save nothing
PUSHL CODE
CALLS #1, SYS\$EXIT
MOVL CODE, R0
RET

: 0952
: 0955
:
: 0956
: 0957

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 02B7

: 781
: 782
: 783
: 784
: 785
0958 1
0959 1
0960 1
0961 1 END
0962 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBAL\$	28	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

CSP
V04-000

Cluster Server Process - Main Routine
CSP\$\$CRASH - Report bug

I 11
16-Sep-1984 01:13:39
14-Sep-1984 13:17:51

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[SYSLOA.SRC]CSP.B32;1 Page 30
(17)

```
: SOWNS          256 NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
: $CODE$        712 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
: $SPLITS       120 NOVEC,NOWRT, RD ,NOEXE,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	21	0	1000	00:01.5

COMMAND QUALIFIERS

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

```
: 786          0963  0
: Size:        712 code + 404 data bytes
: Run Time:    00:13.0
: Elapsed Time: 00:47.7
: Lines/CPU Min: 4458
: Lexemes/CPU-Min: 26847
: Memory Used: 118 pages
: Compilation Complete
```


0393 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

