


```

TTTTTTTTTT  IIIIII  MM      MM  EEEEEEEEE  SSSSSSS  TTTTTTTTTT  AAAAAA  MM      MM  PPPPPPP
TTTTTTTTTT  IIIIII  MM      MM  EEEEEEEEE  SSSSSSS  TTTTTTTTTT  AAAAAA  MM      MM  PPPPPPP
TT          II      MMMM  MMMM  EE          SS      TT          AA      AA  MMMM  MMMM  PP      PP
TT          II      MMMM  MMMM  EE          SS      TT          AA      AA  MMMM  MMMM  PP      PP
TT          II      MM   MM  MM   MM  EE          SS      TT          AA      AA  MM   MM  MM   MM  PP      PP
TT          II      MM   MM  MM   MM  EE          SS      TT          AA      AA  MM   MM  MM   MM  PP      PP
TT          II      MM      MM  EEEEEEE  SSSSSS  TT          AA      AA  MM      MM  PPPPPPP
TT          II      MM      MM  EEEEEEE  SSSSSS  TT          AA      AA  MM      MM  PPPPPPP
TT          II      MM      MM  EE          SS      TT          AAAAAAAAAA  MM      MM  PP
TT          II      MM      MM  EE          SS      TT          AAAAAAAAAA  MM      MM  PP
TT          II      MM      MM  EE          SS      TT          AA      AA  MM      MM  PP
TT          II      MM      MM  EE          SS      TT          AA      AA  MM      MM  PP
TT          II      MM      MM  EE          SS      TT          AA      AA  MM      MM  PP
TT          IIIIII  MM      MM  EEEEEEEEE  SSSSSSS  TT          AA      AA  MM      MM  PP
TT          IIIIII  MM      MM  EEEEEEEEE  SSSSSSS  TT          AA      AA  MM      MM  PP

```

```

LL          IIIIII  SSSSSSS
LL          IIIIII  SSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSS
LL          II      SSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLL  IIIIII  SSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSS

```

```

1 0001 0 MODULE OPC$TIMESTAMP (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 0
6 0006 0 *****
7 0007 0 *
8 0008 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 0 * ALL RIGHTS RESERVED.
11 0011 0 *
12 0012 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 0 * TRANSFERRED.
18 0018 0 *
19 0019 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 0 * CORPORATION.
22 0022 0 *
23 0023 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 0 *
26 0026 0 *
27 0027 0 *****
28 0028 0
29 0029 0 ++
30 0030 0 FACILITY:
31 0031 0
32 0032 0 OPCOM
33 0033 0
34 0034 0 ABSTRACT:
35 0035 0
36 0036 0 This module contains all the various and sundry general
37 0037 0 purpose utility routines used by OPCOM's request handlers.
38 0038 0
39 0039 0 Environment:
40 0040 0
41 0041 0 VAX/VMS operating system.
42 0042 0
43 0043 0 Author:
44 0044 0
45 0045 0 Steven T. Jeffreys
46 0046 0
47 0047 0 Creation date:
48 0048 0
49 0049 0 March 10, 1981
50 0050 0
51 0051 0 Revision history:
52 0052 0
53 0053 0 V03-005 CWH3005 CW Hobbs 25-Jul-1984
54 0054 0 Tune the workset purge algorithm to eliminate purges on
55 0055 0 a quiet OPCOM.
56 0056 0
57 0057 0 V03-004 CWH3169 CW Hobbs 5-May-1984

```

```

: 58 0058 0 : Second pass for cluster-wide OPCOM:
: 59 0059 0 :   - Slow from 15 second timestamps to 5 minute timestamps.
: 60 0060 0 :   - No longer do configures during a timestamp.
: 61 0061 0 :   - Purge the working set on an hourly basis.
: 62 0062 0 :
: 63 0063 0 : V03-003 CWH3003      CW Hobbs      16-Sep-1983
: 64 0064 0 :   Clear ioerr flag at each timestamp
: 65 0065 0 :
: 66 0066 0 : V03-002 CWH3002      CW Hobbs      30-Jul-1983
: 67 0067 0 :   Various and sundry things to make OPCOM distributed
: 68 0068 0 :   across the cluster.
: 69 0069 0 :
: 70 0070 0 : V03-001 STJ3031      Steven T. Jeffreys,  05-Oct-1982
: 71 0071 0 :   - Added the IMPLICITLY_CANCELED routine.
: 72 0072 0 :   - Added the IMPLIED_CANCEL routine.
: 73 0073 0 :   - Added the IMPLIED_DISABLE dummy routine.
: 74 0074 0 :   - Flush the logfile if it has been written to.
: 75 0075 0 :
: 76 0076 0 : --
: 77 0077 0 :
: 78 0078 1 : BEGIN                                ! Start of TIMESTAMP
: 79 0079 1 :
: 80 0080 1 : LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 81 0081 1 : LIBRARY 'LIB$:OPCOMLIB';
: 82 0082 1 :
: 83 0083 1 : FORWARD ROUTINE
: 84 0084 1 :   TIME STAMP      : NOVALUE,      ! Periodic wakeup routine
: 85 0085 1 :   IMPLICITLY_CANCELED, ! Determine if request canceled
: 86 0086 1 :   IMPLIED_CANCEL  : NOVALUE,      ! Perform implicit request cancellation
: 87 0087 1 :   IMPLIED_DISABLE : NOVALUE;      ! Perform implicit operator disable
: 88 0088 1 :
: 89 0089 1 : BUILTIN
: 90 0090 1 :   INSQUE,          ! Insert entry onto a queue
: 91 0091 1 :   REMQUE;          ! Remove entry from a queue
: 92 0092 1 :

```



```

: 151 0150 2 EXTERNAL
: 152 0151 2 LOGFILE_RAB : $bblock, : RAB for operator logfile
: 153 0152 2 OCD_VECTOR : VECTOR, : OCD list heads
: 154 0153 2 GLOBAL_STATUS : BITVECTOR, : Global status bits for OPCOM
: 155 0154 2 NOD_HEAD : VECTOR [?, LONG], : Head of node queue
: 156 0155 2 WAIT_DELTA : $ref_bblock, : Delta time quadword
: 157 0156 2 SYI_SWPOUTPGCNT : LONG, : Swap out page count
: 158 0157 2 LOGTIME_COUNTER : LONG; : Counter for log file timestamp messages
: 159 0158 2
: 160 0159 2 GLOBAL
: 161 0160 2 PURGE_LIMIT : LONG; : Make it easy to find with SDA
: 162 0161 2 : : Purge work set if above this value
: 163 0162 2 OWN
: 164 0163 2 GPGCNT : LONG, : Global pages in working set
: 165 0164 2 PPGCNT : LONG, : Process pages in working set
: 166 0165 2 JPI_WSITEMS : VECTOR [8, LONG] : Item list to get working set items
: 167 0166 2 PRESET ([0] = (jpi$ gpgcnt^16 OR 4),
: 168 0167 2 [1] = GPGCNT,
: 169 0168 2 [2] = 0,
: 170 0169 2 [3] = (jpi$ ppgcnt^16 OR 4),
: 171 0170 2 [4] = PPGCNT,
: 172 0171 2 [5] = 0,
: 173 0172 2 [6] = 0, : End of item list, head of $PURGWS addr desc
: 174 0173 2 [7] = %X'7FFFFFFF'); : End of $PURGWS addr desc
: 175 0174 2
: 176 0175 2 LOCAL
: 177 0176 2 RQST : $ref_bblock, : RQCB (request) data structure
: 178 0177 2 NEXT_RQST : $ref_bblock, : ditto
: 179 0178 2 RQST_COUNT : LONG, : Count of requests in list
: 180 0179 2 NOD : $ref_bblock, : Node data structure
: 181 0180 2 OCD : $ref_bblock, : OCD data structure
: 182 0181 2 NEXT_OCD : $ref_bblock, : ditto
: 183 0182 2 OCD_COUNT : LONG, : Count of OCDs in list
: 184 0183 2 STATUS : LONG;
: 185 0184 2
: 186 0185 2 :
: 187 0186 2 : If shutdown is pending, then do nothing.
: 188 0187 2
: 189 0188 2 IF .GLOBAL_STATUS [GBLSTS_K_SHUTDOWN_PENDING]
: 190 0189 2 THEN
: 191 0190 2 BEGIN
: 192 0191 2 GLOBAL_STATUS [GBLSTS_K_TIMESTAMP_PENDING] = FALSE;
: 193 0192 2 RETURN;
: 194 0193 2 END;
: 195 0194 2
: 196 0195 2 : Set GBLSTS_K_TIMESTAMP_PENDING. If OPCOM is busy, then return.
: 197 0196 2 : If not, then set GBLSTS_K_BUSY to prevent another timestamp AST from arriving.
: 198 0197 2
: 199 0198 2 GLOBAL_STATUS [GBLSTS_K_TIMESTAMP_PENDING] = TRUE;
: 200 0199 2 IF .GLOBAL_STATUS [GBLSTS_K_BUSY]
: 201 0200 2 THEN
: 202 0201 2 RETURN;
: 203 0202 2 GLOBAL_STATUS [GBLSTS_K_BUSY] = TRUE;
: 204 0203 2
: 205 0204 2 :
: 206 0205 2 : Every twelve timestamps (once an hour), stamp the log file. Also, since we might
: 207 0206 2 : have a lot of garbage sitting in memory, flush the working set so that we do not

```

```

208 0207 2 : eat up unnecessary pages on small systems.
209 0208 2
210 0209 2 IF (LOGTIME_COUNTER = .LOGTIME_COUNTER + 1) GEQ 12
211 0210 2 THEN
212 0211 2 BEGIN
213 0212 2 +
214 0213 2 | Start of 60 minute timestamp
215 0214 2 -
216 0215 2 LOCAL
217 0216 2 MSGVEC : VECTOR [2, LONG], ! Temporary vector for message
218 0217 2 LOG_RQCB : $ref_block;
219 0218 2 LOGTIME_COUNTER = 0;
220 0219 2 IF ALLOCATE_DS (RQCB_K_TYPE, LOG_RQCB)
221 0220 2 THEN
222 0221 2 BEGIN
223 0222 2 MSGVEC [0] = OPCS_LOGTIME;
224 0223 2 MSGVEC [1] = 0;
225 0224 2 FORMAT MESSAGE (.LOG_RQCB, MSGVEC);
226 0225 2 LOG MESSAGE (.LOG_RQCB);
227 0226 2 DEALLOCATE_RQCB (.LOG_RQCB);
228 0227 2 END;
229 0228 2
230 0229 2 | Flush the working set, but first check to make sure that we are big enough to need it
231 0230 2 | Note also that by flushing before the 5 minute section, we will most likely fault in
232 0231 2 | the code and data needed by the timestamp from the lists, rather than doing real I/O.
233 0232 2
234 0233 2 IF NOT (STATUS = $GETJPI (ITMLST=JPI_WSITEMS))
235 0234 2 THEN
236 0235 2 $signal_stop (.STATUS);
237 0236 2 IF .PPGCNT+.GPGCNT GTR .PURGE_LIMIT
238 0237 2 THEN
239 0238 2 BEGIN
240 0239 2 PURGE_LIMIT = 0; ! Reset so we will recalculate what we need
241 0240 2 $PURGWS (INADR=JPI_WSITEMS[6]); ! Reuse a longword of the item list
242 0241 2 END;
243 0242 2 +
244 0243 2 | End of 60 minute
245 0244 2 -
246 0245 2 END;
247 0246 2
248 0247 2 +
249 0248 2 | Start of 5 minute timestamp
250 0249 2 -
251 0250 2
252 0251 2
253 0252 2 | For each request outstanding, notify all interested operators.
254 0253 2
255 0254 2 | Before notifying the interested operators, check to see if the request
256 0255 2 | has been implicitly canceled. If so, insert it on a special queue for
257 0256 2 | processing later in this routine.
258 0257 2
259 0258 2 | Also note that as this is happening, implicitly disabled operators are
260 0259 2 | being processed. They too will be removed from the data base later in
261 0260 2 | this routine.
262 0261 2
263 0262 2
264 0263 2 INCR I FROM MIN_SCOPE TO MAX_SCOPE DO

```

```

265 0264 BEGIN
266 0265
267 0266   For each each class of operator (SYSTEM, GROUP, USER) ...
268 0267
269 0268 NEXT_OCD = .OCD_VECTOR [(I-1)*2];           ! Get first OCD in list
270 0269 INCR J FROM 1 TO .OCD_VECTOR [(I-1)*2+1] DO
271 0270 BEGIN
272 0271   For each OCD in the operator class list...
273 0272
274 0273   OCD = .NEXT_OCD;                          ! Get current OCD address
275 0274   NEXT_OCD = .OCD [OCD_L_FLINK];          ! Get next OCD address
276 0275   NEXT_RQST = .OCD [OCD_L_RQSTFLINK];    ! Get first request address
277 0276   INCR K FROM 1 TO .OCD [OCD_W_RQSTCOUNT] DO
278 0277 BEGIN
279 0278   For each request in the OCD list...
280 0279
281 0280   RQST = .NEXT_RQST;                        ! Get current request address
282 0281   NEXT_RQST = .RQST [RQCB_L_FLINK];      ! Get next request address
283 0282   IF NOT IMPLICITLY_CANCELED (.RQST)
284 0283   THEN
285 0284     The reply mailbox exists. Inform operators of the request.
286 0285     NOTIFY_LISTED_OPERATORS (.RQST)
287 0286   END;
288 0287 END;
289 0288 END;
290 0289
291 0290
292 0291
293 0292
294 0293
295 0294   After sweeping through the data base, we may have discovered some
296 0295   implicitly canceled requests and implicitly disabled operators.
297 0296   Process them now. The requests should be done first, as yet more
298 0297   implicitly disabled operators may turn up.
299 0298
300 0299 IMPLIED_CANCEL ();
301 0300 IMPLIED_DISABLE ();
302 0301
303 0302   Make a scan through the node database
304 0303
305 0304 NOD = .NOD_HEAD [0];
306 0305 WHILE .NOD_NEQ NOD_HEAD [0]
307 0306 DO
308 0307 BEGIN
309 0308
310 0309   Clear the error message flag. This limits the rate of OPCS_CLUSCOMM error messages to
311 0310   one per five minutes.
312 0311
313 0312 NOD [NOD_V_IOERR_DISPLAYED] = FALSE;
314 0313
315 0314   If we have any nodes in "START" state, then request an acknowledgement from them.
316 0315
317 0316 IF .NOD [NOD_B_STATE] EQL NOD_K_STATE_START
318 0317 THEN
319 0318 BEGIN
320 0319   NOD [NOD_V_ACK_PEND] = FALSE;           ! Clear so that we can
321 0320   CLUSMSG_ACR_PLEASE (.NOD);           ! request an acknowledgement

```



```

322 0321 1
323 0322 2
324 0323 3
325 0324 4
326 0325 5
327 0326 6
328 0327 7
329 0328 8
330 0329 9
331 0330 10
332 0331 11
333 0332 12
334 0333 13
335 0334 14
336 0335 15
337 0336 16
338 0337 17
339 0338 18
340 0339 19
341 0340 20
342 0341 21
343 0342 22
344 0343 23
345 0344 24
346 0345 25
347 0346 26
348 0347 27
349 0348 28
350 0349 29
351 0350 30
352 0351 31
353 0352 32
354 0353 33
355 0354 34
356 0355 35
357 0356 36
358 0357 37
359 0358 38
360 0359 39
361 0360 40
362 0361 41
363 0362 42
364 0363 43
365 0364 44
366 0365 45
367 0366 46
368 0367 47
369 0368 48
370 0369 49

```

```

      END;
      NOD = .NOD [NOD_L_FLINK];
      END;

      If the operator logfile was written to since the last timestamp operation,
      flush the contents of the RMS buffers to the disk. This also has the effect
      of writing the file header, so the information is not lost in the event of
      a system crash. This is necessary because the log file is kept open until
      explicitly closed via REPLY/[NO]LOG.

      IF .GLOBAL_STATUS [GBLSTS_K_FLUSH_PENDING]
      THEN
      BEGIN
      GLOBAL_STATUS [GBLSTS_K_FLUSH_PENDING] = FALSE;
      $FLUSH (RAB = LOGFILE_RAB);
      END;

      If we purged the working set on this pass, then save the size we have now.
      This lets us react to peaks in working set use, without a lot of faults
      during periods of non-activity.

      IF .PURGE_LIMIT EQL 0
      THEN
      BEGIN
      REGISTER
      SWAPO, LIMIT;
      IF NOT (STATUS = $GETJPI (ITMLST=JPI_WSITEMS))
      THEN
      $signal_stop (.STATUS);

      Set new value to 10 more pages than we are currently using, but no lower
      than swap-out-page-count and no higher than 3 times swap-out-page-count.

      SWAPO = .SYI SWPOUTPGCNT;           ! Get it into a register
      LIMIT = MAX (.PPGCNT+.GPGCNT+10, .SWAPO); ! Limit is larger of swapo and 10 more than current
      SWAPO = 3 * .SWAPO;                 ! Compute the max
      PURGE_LIMIT = MIN (.SWAPO, .LIMIT); ! Actual limit is smaller of the two
      END;

      Queue another timer ast.

      GLOBAL_STATUS [GBLSTS_K_BUSY] = FALSE;
      GLOBAL_STATUS [GBLSTS_K_TIMESTAMP_PENDING] = FALSE;
      IF NOT (STATUS = $SETIMR (EFN = EFN_K_TIME_STAMP, DAYTIM = WAIT_DELTA, ASTADR = TIME_STAMP))
      THEN
      $signal_stop (.STATUS);

      ! End of TIME_STAMP

```

```

.TITLE OPC$TIMESTAMP
.IDENT \V04-000\

.PSECT $OWNS,NOEXE,2

```

```
00000 GPGCNT: .BLKB 4
```

```

00004 PPGCNT: .BLKB 4
030C0004 00008 JPI_WSITEMS:
          .LONG 51118084
00000000' 0000C .ADDRESS GPGCNT
030D0004 00000000 00010 .LONG 0, 51183620
          00000000' 00018 .ADDRESS PPGCNT
7FFFFFFF 00000000 00000000 0001C .LONG 0, 0, 2147483647

          .PSECT $GLOBAL$,NOEXE,2

00000 PURGE_LIMIT::
          .BLKB 4

          .EXTRN ALLOCATE_DS, CLUSMSG_ACK_PLEASE
          .EXTRN CLUSMSG_STATE_SEND
          .EXTRN DEALLOCATE_RQCB
          .EXTRN FORMAT_MESSAGE, LOG_MESSAGE
          .EXTRN NOTIFY_LISTED_OPERATORS
          .EXTRN RQCB_K_TYPE, MIN_SCOPE
          .EXTRN MAX_SCOPE, LOGFICE_RAB
          .EXTRN OCD_VECTOR, GLOBAL_STATUS
          .EXTRN NOD_HEAD, WAIT_DELTA
          .EXTRN SYI_SWPOUTPGCNT
          .EXTRN LOGTIME_COUNTER
          .EXTRN SYSSGETJPI, LIB$STOP
          .EXTRN SYSSPURGWS, SYSSFLUSH
          .EXTRN SYSSSETIMR

          .PSECT $CODE$,NOWRT,2

          OFFC 00000 .ENTRY TIME STAMP, Save R2,R3,R4,R5,R6,R7,R8,R9,-
          .R10,R11
          SE 06 0000G 0C C2 00002 00002
          06 0000G CF 0000G CF E9 00005 00005
          0000G CF 20 8A 0000A 0000A
          04 0000F 04 0000F 0000F
01 0000G CF 20 88 00010 1$: 00010
0000G CF 06 E1 00015 00015
          04 0001B 04 0001B 0001B
50 0000G CF 4) 8F 88 0001C 2$: 0001C
0000G CF 01 C1 00022 00022
0000G CF 50 D0 00028 00028
          OC 50 D1 0002D 0002D
          72 19 00030 00030
          0000G CF D4 00032 00032
          5E DD 00036 00036
          00000000G 8F DD 00038 00038
0000G CF 02 FB 0003E 0003E
          24 50 E9 00043 00043
          04 AE 00058099 8F D0 00046 00046
          08 AE D4 0004E 0004E
          04 AE 9F 00051 00051
          04 AE DD 00054 00054
0000G CF 02 FB 00057 00057
          6E DD 0005C 0005C
0000G CF 01 FB 0005E 0005E
          6E DD 00063 00063

          SUBL2 #12, SP
          BLBC GLOBAL_STATUS, 1$
          BICB2 #32, GLOBAL_STATUS
          RET
          BISB2 #32, GLOBAL_STATUS
          BBC #6, GLOBAL_STATUS, 2$
          RET
          BISB2 #64, GLOBAL_STATUS
          ADDL3 #1, LOGTIME_COUNTER, R0
          MOVL R0, LOGTIME_COUNTER
          Cmpl R0, #12
          BLSS 5$
          CLRL LOGTIME_COUNTER
          PUSHL SP
          PUSHL #RQCB_K_TYPE
          CALLS #2, ALLOCATE_DS
          BLBC R0, 3$
          MOVL #360601, MSGVEC
          CLRL MSGVEC+4
          PUSHAB MSGVEC
          PUSHL LOG_RQCB
          CALLS #2, -FORMAT_MESSAGE
          PUSHL LOG_RQCB
          CALLS #1, -LOG_MESSAGE
          PUSHL LOG_RQCB

```

| | | | | | | | | | |
|----|-----------|----|-----------|------|----|-------|--------------|-----------------------------|------|
| | 0000G | CF | | 01 | FB | 00065 | CALLS | #1, DEALLOCATE_RQCB | |
| | | | | 7E | 7C | 0006A | 3\$: CLRQ | -(SP) | 0233 |
| | | | | 7E | D4 | 0006C | CLRL | -(SP) | |
| | | | 0000' | CF | 9F | 0006E | PUSHAB | JPI WSITEMS | |
| | | | | 7E | 7C | 00072 | CLRQ | -(SP) | |
| | | | | 7E | D4 | 00074 | CLRL | -(SP) | |
| | 00000000G | 00 | | 07 | FB | 00076 | CALLS | #7, SYSS\$GETJPI | |
| | | 5B | | 50 | D0 | 0007D | MOVL | RO, STATUS | |
| | | 03 | | 5B | E8 | 00080 | BLBS | STATUS, 4\$ | |
| | | | | 0125 | 31 | 00083 | BRW | 19\$ | |
| 50 | 0000' | CF | 0000' | CF | C1 | 00086 | 4\$: ADDL3 | GPGCNT, PPGCNT, RO | 0236 |
| | 0000' | CF | | 50 | D1 | 0008E | CMPL | RO, PURGE_LIMIT | |
| | | | | 0F | 15 | 00093 | BLEQ | 5\$ | |
| | | | 0000' | CF | D4 | 00095 | CLRL | PURGE_LIMIT | 0239 |
| | | | 0000' | CF | 9F | 00099 | PUSHAB | JPI WSITEMS+24 | 0240 |
| | 00000000G | 00 | | 01 | FB | 0009D | CALLS | #1, -SYSS\$PURGWS | |
| 52 | 00000000G | 8F | | 01 | C3 | 00CA4 | 5\$: SUBL3 | #1, #MIN_SCOPE, I | 0263 |
| | | | | 45 | 11 | 000AC | BRB | 11\$ | |
| 50 | | 52 | | 01 | 78 | 000AE | 6\$: ASHL | #1, I, RO | 0268 |
| | | 58 | 0000GCF | 40 | D0 | 000B2 | MOVL | OCD_VECTOR-8[RO], NEXT_OCD | |
| | | 5A | 0000GCF | 40 | D0 | 000B8 | MOVL | OCD_VECTOR-4[RO], R10 | 0269 |
| | | | | 55 | D4 | 000BE | CLRL | J | |
| | | | | 2D | 11 | 000C0 | BRB | 10\$ | |
| | | 53 | | 58 | D0 | 000C2 | 7\$: MOVL | NEXT_OCD, OCD | 0274 |
| | | 58 | | 63 | D0 | 000C5 | MOVL | (OCD), NEXT_OCD | 0275 |
| | | 59 | 3C | A3 | D0 | 000C8 | MOVL | 60(OCD), NEXT_RQST | 0276 |
| | | 57 | 3A | A3 | 3C | 000CC | MOVZWL | 58(OCD), R7 | 0277 |
| | | | | 54 | D4 | 000D0 | CLRL | K | |
| | | | | 17 | 11 | 000D2 | BRB | 9\$ | |
| | | 56 | | 59 | D0 | 000D4 | 8\$: MOVL | NEXT_RQST, RQST | 0282 |
| | | 59 | | 66 | D0 | 000D7 | MOVL | (RQST), NEXT_RQST | 0283 |
| | | | | 56 | DD | 000DA | PUSHL | RQST | 0284 |
| | 0000V | CF | | 01 | FB | 000DC | CALLS | #1, IMPLICITLY_CANCELED | |
| | | 07 | | 50 | E8 | 000E1 | BLBS | RO, 9\$ | |
| | | | | 56 | DD | 000E4 | PUSHL | RQST | 0289 |
| | 0000G | CF | | 01 | FB | 000E6 | CALLS | #1, NOTIFY_LISTED_OPERATORS | |
| E5 | | 54 | | 57 | F3 | 000EB | 9\$: AOBLEQ | R7, K, 8\$ | 0284 |
| CF | | 55 | | 5A | F3 | 000EF | 10\$: AOBLEQ | R10, J, 7\$ | 0269 |
| B3 | | 52 | 00000000G | 8F | F3 | 000F3 | 11\$: AOBLEQ | #MAX_SCOPE, I, 6\$ | 0263 |
| | 0000V | CF | | 00 | FB | 000FB | CALLS | #0, IMPLIED_CANCEL | 0299 |
| | 0000V | CF | | 00 | FB | 00100 | CALLS | #0, IMPLIED_DISABLE | 0300 |
| | | 53 | 0000G | CF | D0 | 00105 | MOVL | NOD_HEAD, NOD | 0304 |
| | | 50 | 0000G | CF | 9E | 0010A | 12\$: MOVAB | NOD_HEAD, RO | 0305 |
| | | 50 | | 53 | D1 | 0010F | CMPL | NOD, RO | |
| | | | | 1A | 13 | 00112 | BEQL | 14\$ | |
| | 2A | A3 | | 04 | 8A | 00114 | BICB2 | #4, 42(NOD) | 0312 |
| | | 02 | 22 | A3 | 91 | 00118 | CMPB | 34(NOD), #2 | 0316 |
| | | | | 0B | 12 | 0011C | BNEQ | 13\$ | |
| | 2A | A3 | | 01 | 8A | 0011E | BICB2 | #1, 42(NOD) | 0319 |
| | | | | 53 | DD | 00122 | PUSHL | NOD | 0320 |
| | 0000G | CF | | 01 | FB | 00124 | CALLS | #1, CLUSMSG_ACK_PLEASE | |
| | | 53 | | 63 | D0 | 00129 | 13\$: MOVL | (NOD), NOD | 0322 |
| | | | | DC | 11 | 0012C | BRB | 12\$ | 0305 |
| | | | 0000G | CF | 95 | 0012E | 14\$: TSTB | GLOBAL_STATUS | 0332 |
| | | | | 11 | 18 | 00132 | BGEQ | 15\$ | |
| | 0000G | CF | 80 | 8F | 8A | 00134 | BICB2 | #128, GLOBAL_STATUS | 0335 |
| | | | 0000G | CF | 9F | 0013A | PUSHAB | LOGFILE_RAB | 0336 |

| | | | | | | | |
|-----------|-------|-------|----------------|--------|------------------------|---|------|
| 00000000G | 00 | 0000' | 01 FB 0013E | CALLS | #1, SYSS\$FLUSH | : | |
| | | | CF D5 00145 | TSTL | PURGE_LIMIT | : | 0343 |
| | | | 41 12 00149 | BNEQ | 18\$ | : | |
| | | | 7E 7C 0014B | CLRQ | -(SP) | : | 0348 |
| | | | 7E D4 0014D | CLRL | -(SP) | : | |
| | | 0000' | CF 9F 0014F | PUSHAB | JPI WSITEMS | : | |
| | | | 7E 7C 00153 | CLRQ | -(SP) | : | |
| | | | 7E D4 00155 | CLRL | -(SP) | : | |
| 00000000G | 00 | | 07 FB 00157 | CALLS | #7, SYSS\$GETJPI | : | |
| | 5B | | 50 D0 0015E | MOVL | R0, STATUS | : | |
| | 47 | | 5B E9 00161 | BLBC | STATUS, 19\$ | : | |
| 51 | 0000' | 0000G | CF D0 00164 | MOVL | SYI SWPOUTPGCNT, SWAPO | : | 0355 |
| | | 0000' | CF C1 00169 | ADDL3 | PGCNT, PPGCNT, R1 | : | 0356 |
| | | | 51 0A C0 00171 | ADDL2 | #10, R1 | : | |
| | | | 50 51 D1 00174 | CMPL | R1, SWAPO | : | |
| | | | 03 18 00177 | BGEQ | 16\$ | : | |
| | | | 50 D0 00179 | MOVL | SWAPO, R1 | : | |
| | | | 50 03 C4 0017C | MULL2 | #3, SWAPO | : | 0357 |
| | | | 50 D1 0017F | CMPL | R0, LIMIT | : | 0358 |
| | | | 03 15 00182 | BLEQ | 17\$ | : | |
| | | | 51 D0 00184 | MOVL | LIMIT, R0 | : | |
| | 0000' | | 50 D0 00187 | MOVL | R0, PURGE_LIMIT | : | |
| | 0000G | CF | 60 8F 8A 0018C | BICB2 | #96, GLOBAL_STATUS | : | 0364 |
| | | | 7E D4 00192 | CLRL | -(SP) | : | 0365 |
| | | FE68 | CF 9F 00194 | PUSHAB | TIME_STAMP | : | |
| | | 0000G | CF 9F 00198 | PUSHAB | WAIT_DELTA | : | |
| | | | 04 DD 0019C | PUSHL | #4 | : | |
| 00000000G | 00 | | 04 FB 0019E | CALLS | #4, SYSS\$SETIMR | : | |
| | 5B | | 50 D0 001A5 | MOVL | R0, STATUS | : | |
| | 09 | | 5B E8 001A6 | BLBS | STATUS, 20\$ | : | |
| | | | 5B DD 001AB | PUSHL | STATUS | : | 0367 |
| 00000000G | 00 | | 01 FB 001AD | CALLS | #1, LIB\$STOP | : | |
| | | | 04 001B4 | 20\$: | RET | : | 0369 |

; Routine Size: 437 bytes, Routine Base: \$CODE\$ + 0000

```

372 0370 1 GLOBAL ROUTINE IMPLICITLY_CANCELED (RQST) =
373 0371 1
374 0372 1 |**
375 0373 1 | Functional description:
376 0374 1 |
377 0375 1 |     Check a given request to see if it has been implicitly canceled.
378 0376 1 |     An implicit cancelation is defined as the requestor deleting the
379 0377 1 |     reply mailbox without first having sent an explicit request cancelation
380 0378 1 |     message to OPCOM.
381 0379 1 |
382 0380 1 | Input:
383 0381 1 |
384 0382 1 |     RQST      : address of a request control block
385 0383 1 |
386 0384 1 | Implicit Input:
387 0385 1 |
388 0386 1 |     None.
389 0387 1 |
390 0388 1 | Output:
391 0389 1 |
392 0390 1 |     None.
393 0391 1 |
394 0392 1 | Implicit output:
395 0393 1 |
396 0394 1 |     None.
397 0395 1 |
398 0396 1 | Side effects:
399 0397 1 |
400 0398 1 |     If the request has been implicitly canceled, it will be inserted
401 0399 1 |     into a queue of canceled requests.  The queue will be processed later.
402 0400 1 |
403 0401 1 | Routine value:
404 0402 1 |
405 0403 1 |     TRUE      : the request has been implicitly canceled
406 0404 1 |     FALSE     : the request is still active
407 0405 1 | --
408 0406 1 |
409 0407 2 BEGIN                                ! Start of IMPLICITLY_CANCELED
410 0408 2
411 0409 2 MAP
412 0410 2     RQST          : $ref_bblock;          ! Request control block
413 0411 2
414 0412 2 EXTERNAL ROUTINE
415 0413 2     CLUSUTIL_SYSTEMID_EQUAL : JSB_ROR1;
416 0414 2
417 0415 2 EXTERNAL
418 0416 2     GLOBAL STATUS : BITVECTOR [32],
419 0417 2     CANCELED_RQST_Q : VECTOR,           ! List head of canceled requests
420 0418 2     LCL_NOD        : $ref_bblock,
421 0419 2     MBX_FAO        : $bblock;          ! FAO control string
422 0420 2
423 0421 2 LOCAL
424 0422 2     MBX_NAME       : $bblock [MAX_DEV_NAM], ! Mailbox device name buffer
425 0423 2     MBX_DESC       : $desc_block,         ! Mailbox device name descriptor
426 0424 2     DEV_CHAR       : $bblock [DIB$K_LENGTH], ! Mailbox dev. char. buffer
427 0425 2     CHAR_DESC      : $desc_block;         ! Mailbox dev. char. descriptor
428 0426 2

```

```

429 0427 2 |
430 0428 2 | Do not implicitly cancel requests from other nodes
431 0429 2 |
432 0430 2 | IF .GLOBAL_STATUS [GBLSTS_K_IN_VAXcluster]
433 0431 2 | THEN
434 0432 2 |   IF NOT CLUSUTIL_SYSTEMID_EQUAL (RQST [RQCB_T_SYSTEMID], LCL_NOD [NOD_T_NODE_SYSTEMID])
435 0433 2 |   THEN
436 0434 2 |     RETURN FALSE;           ! Not disabled
437 0435 2 |
438 0436 2 |
439 0437 2 | Check to see if the request has been implicitly canceled.
440 0438 2 | The simplest way to do this is to attempt to get the device
441 0439 2 | characteristics.  If the device no longer exists, then assume
442 0440 2 | the user is no longer interested in the request.  First format
443 0441 2 | the mailbox name from the information in the RQCB.
444 0442 2 |
445 0443 2 |
446 0444 2 | MBX_DESC [DSC$W_LENGTH] = MAX_DEV_NAM; ! Create a descriptor
447 0445 2 | MBX_DESC [DSC$B_DTYPE] = 0;
448 0446 2 | MBX_DESC [DSC$B_CLASS] = 0;
449 0447 2 | MBX_DESC [DSC$A_POINTER] = MBX_NAME;
450 0448 2 | $FAO (MBX_FAO, MBX_DESC, MBX_DESC, .RQST [RQCB_W_REPLYMBX]);
451 0449 2 | CHAR_DESC [DSC$W_LENGTH] = DIB$K_LENGTH; ! Create a descriptor
452 0450 2 | CHAR_DESC [DSC$B_DTYPE] = 0;
453 0451 2 | CHAR_DESC [DSC$B_CLASS] = 0;
454 0452 2 | CHAR_DESC [DSC$A_POINTER] = DEV_CHAR;
455 0453 2 | IF ($GETDEV (DEVNAM=MBX_DESC, PRIBUF=CHAR_DESC))
456 0454 2 | THEN
457 0455 2 |
458 0456 2 |   The reply mailbox still exists.
459 0457 2 |
460 0458 2 |   RETURN FALSE
461 0459 2 | ELSE
462 0460 2 | BEGIN
463 0461 2 |
464 0462 2 |   The reply mailbox no longer exists.  Assume request canceled.
465 0463 2 |
466 0464 3 |   RQST [RQSTS_V_IMPCANCEL] = TRUE;
467 0465 3 |   INSQUE (RQST [RQCB_L_DSBLFLINK], CANCELED_RQST_Q);
468 0466 3 |   RETURN TRUE;
469 0467 2 |   END;
470 0468 2 |
471 0469 1 | END;

```

! End of IMPLICITLY_CANCELED

```

.EXTRN CLUSUTIL_SYSTEMID_EQUAL
.EXTRN CANCELED_RQST_Q
.EXTRN LCL_NOD, MBX_FAO
.EXTRN SYS$FAO, SYS$GETDEV

```

```

                    0004 0000
SE      FF3C      CE 9E 0002
15      0000G    CF E9 0007
51      0000G    CF 0000050 8F C1 000C
50      04      AC      1C C1 0016
                    0000G 30 001B

```

```

.ENTRY IMPLICITLY_CANCELED, Save R2
MOVAB  -196(SP), SP
BLBC   GLOBAL_STATUS+1, 1$
ADDL3  #80, LCL_NOD, R1
ADDL3  #28, RQST, R0
BSBW   CLUSUTIL_SYSTEMID_EQUAL

```

```

: 0370
: 0430
: 0432
:

```

| | | | | | | | | | | | |
|-----------|----|------|-------|----|-------|-------|------|--------|--------------------------|--|------|
| | 7C | 4F | | 50 | E9 | 0001E | | BLBC | R0, 2\$ | | |
| | BC | AE | 40 | 8F | 9A | 00021 | 1\$: | MOVZBL | #64, MBX_DESC | | 0444 |
| | | AD | C0 | AD | 9E | 00026 | | MOVAB | MBX_NAME, MBX_DESC+4 | | 0447 |
| | | 52 | 04 | AC | D0 | 0002B | | MOVL | RQST, R2 | | 0448 |
| | | 7E | 2E | A2 | 3C | 0002F | | MOVZWL | 46(R2), -(SP) | | |
| | | | B8 | AD | 9F | 00033 | | PUSHAB | MBX_DESC | | |
| | | | B8 | AD | 9F | 00036 | | PUSHAB | MBX_DESC | | |
| | | | 0000G | CF | 9F | 00039 | | PUSHAB | MBX_FAO | | |
| 00000000G | 00 | | | 04 | FB | 0003D | | CALLS | #4, -SYS\$FAO | | |
| | 6E | | 74 | 8F | 9A | 00044 | | MOVZBL | #116, CHAR_DESC | | 0449 |
| 04 | AE | | 08 | AE | 9E | 00048 | | MOVAB | DEV CHAR, CHAR_DESC+4 | | 0452 |
| | | | | 7E | 7C | 0004D | | CLRQ | -(SP) | | 0453 |
| | | | 08 | AE | 9F | 0004F | | PUSHAB | CHAR_DESC | | |
| | | | | 7E | D4 | 00052 | | CLRL | -(SP) | | |
| | | | B8 | AD | 9F | 00054 | | PUSHAB | MBX_DESC | | |
| 00000000G | 00 | | | 05 | FB | 00057 | | CALLS | #5, -SYS\$GETDEV | | |
| | 0F | | | 50 | E8 | 0005E | | BLBS | R0, 2\$ | | |
| 2A | A2 | | | 01 | 88 | 00061 | | BISB2 | #1, 42(R2) | | 0464 |
| 0000G | CF | 008C | | C2 | 0E | 00065 | | INSQUE | 140(R2), CANCELED_RQST_Q | | 0465 |
| | 50 | | | 01 | D0 | 0006C | | MOVL | #1, R0 | | 0466 |
| | | | | | 04 | 0006F | | RET | | | 0460 |
| | | | | 50 | D4 | 00070 | 2\$: | CLRL | R0 | | 0469 |
| | | | | 04 | 00072 | | | RET | | | |

; Routine Size: 115 bytes, Routine Base: \$CODE\$ + 01B5

```

473 0470 1 GLOBAL ROUTINE IMPLIED_CANCEL : NOVALUE =
474 0471 1
475 0472 1  +-+
476 0473 1  Functional description:
477 0474 1
478 0475 1      For all requests on the canceled request queue, create a
479 0476 1      cancellation message from the information in the request
480 0477 1      control block, and CALL the request cancellation handler
481 0478 1      as if the user had sent the cancellation message.
482 0479 1
483 0480 1  Input:
484 0481 1
485 0482 1      None.
486 0483 1
487 0484 1  Implicit Input:
488 0485 1
489 0486 1      CANCELED_RQST_Q : The list head of all implicitly canceled requests.
490 0487 1
491 0488 1  Output:
492 0489 1
493 0490 1      None.
494 0491 1
495 0492 1  Implicit output:
496 0493 1
497 0494 1      None.
498 0495 1
499 0496 1  Side effects:
500 0497 1
501 0498 1      All interested operators will be notified of the canceled requests.
502 0499 1      As this is done, implicitly disabled operators may be discovered.
503 0500 1      Those operators will be placed on the implicit disable queue and
504 0501 1      be processed later.
505 0502 1
506 0503 1  Routine value:
507 0504 1
508 0505 1      None.
509 0506 1  --
510 0507 1
511 0508 2 BEGIN                                ! Start of IMPLIED_CANCEL
512 0509 2
513 0510 2 EXTERNAL ROUTINE
514 0511 2     CNCL_HANDLER      : NOVALUE,          ! Old CANCEL message handler
515 0512 2     NOTIFY_LISTED_OPERATORS;          ! Notify a list of operators
516 0513 2
517 0514 2 EXTERNAL
518 0515 2     CANCELED_RQST_Q : VECTOR;            ! List head of canceled requests
519 0516 2
520 0517 2 LITERAL
521 0518 2     MSG_HDR_SIZE      = ($BYTEOFFSET(RQCB_B_RQSTCODE) - $BYTEOFFSET(RQCB_W_MSGTYPE)),
522 0519 2     OLD_MSG_SIZE      = 8,
523 0520 2     MSG_BUF_SIZE     = MSG_HDR_SIZE + OLD_MSG_SIZE;
524 0521 2
525 0522 2 MACRO
526 0523 2     REQUEST_TYPE     = MSG_HDR_SIZE, 0, 8, 0%;
527 0524 2     TARGET_MASK     = MSG_HDR_SIZE+1, 0, 24, 0%;
528 0525 2     REQUEST_ID      = MSG_HDR_SIZE+4, 0, 32, 0%;
529 0526 2

```



```

: 530      0527 2 LOCAL
: 531      0528 2 CANCEL_MSG_BUF : $bblock [MSG_BUF_SIZE], ! CANCEL request message buffer
: 532      0529 2 CANCEL_MSG_DESC : $desc_block, ! CANCEL request descriptor
: 533      0530 2 RQST : $ref_bblock; ! Request control block
: 534      0531 2
: 535      0532 2
: 536      0533 2
: 537      0534 2 Create the message buffer descriptor. We need do this only once.
: 538      0535 2
: 539      0536 2
: 540      0537 2 CANCEL_MSG_DESC [DSC$W_LENGTH] = MSG_BUF_SIZE;
: 541      0538 2 CANCEL_MSG_DESC [DSC$B_DTYPE] = 0;
: 542      0539 2 CANCEL_MSG_DESC [DSC$B_CLASS] = 0;
: 543      0540 2 CANCEL_MSG_DESC [DSC$A_POINTER] = CANCEL_MSG_BUF;
: 544      0541 2
: 545      0542 2
: 546      0543 2 For all requests on the queue, create a cancel message
: 547      0544 2 (old format) and call the cancel request handler.
: 548      0545 2
: 549      0546 2
: 550      0547 2 WHILE NOT REMQUE (.CANCELED_RQST_Q, RQST) DO
: 551      0548 2 BEGIN
: 552      0549 2 RQST = .RQST - ($BYTEOFFSET(RQCB_L_DSBLFLINK) - $BYTEOFFSET(RQCB_L_FLINK));
: 553      0550 2 CH$MOVE (MSG_HDR_SIZE, RQST [RQCB_Q_MSGTYPE], CANCEL_MSG_BUF);
: 554      0551 2 CANCEL_MSG_BUF [REQUEST_TYPE] = OPC$RQ_CANCEL;
: 555      0552 2 CANCEL_MSG_BUF [TARGET_MASK] = .RQST [RQCB_L_ATTNUMASK1];
: 556      0553 2 CANCEL_MSG_BUF [REQUEST_ID] = .RQST [RQCB_L_RQSTID];
: 557      0554 2 CNCL_HANDLER (CANCEL_MSG_DESC);
: 558      0555 2 END;
: 559      0556 2
: 560      0557 1 END;

```

! End of IMPLIED_CANCEL

.EXTRN CNCL_HANDLER

| | | | | | | |
|----|----|----|-------|-------------------|--------------------------------------------|--------|
| | | | | 007C 0000 | .ENTRY IMPLIED_CANCEL, Save R2,R3,R4,R5,R6 | : 0470 |
| | | | SE | 34 C2 00002 | SUBL2 #52, SP | : 0537 |
| | | | | 2E DD 00005 | PUSHL #46 | : 0540 |
| | 04 | AE | 08 | AE 9E 00007 | MOVAB CANCEL_MSG_BUF, CANCEL_MSG_DESC+4 | : 0547 |
| | | 56 | 0000G | DF 0F 0000C 1\$: | REMQUE @CANCELED_RQST_Q, RQST | : 0549 |
| | | | | 24 1D 00011 | BVS 2\$ | : 0550 |
| | | | 56 | FF74 C6 9E 00013 | MOVAB -140(R6), RQST | : 0551 |
| | 08 | AE | 2C | A6 26 28 00018 | MOV C3 #38, 44(RQST), CANCEL_MSG_BUF | : 0552 |
| | | | 2E | AE 05 90 0001E | MOV B #5, CANCEL_MSG_BUF+38 | : 0553 |
| 2F | AE | | 18 | 00 5C A6 F0 00022 | INSV 92(RQST), #0, #24, CANCEL_MSG_BUF+39 | : 0554 |
| | | | 32 | AE 64 A6 D0 00029 | MOVL 100(RQST), CANCEL_MSG_BUF+42 | : 0557 |
| | | | | 5E DD 0002E | PUSHL SP | |
| | | | 0000G | CF 01 FB 00030 | CALLS #1, CNCL_HANDLER | |
| | | | | D5 11 00035 | BRB 1\$ | : 0547 |
| | | | | 04 00037 2\$: | RET | : 0557 |

: Routine Size: 56 bytes, Routine Base: \$CODE\$ + 0228

```

: 562 0558 1 GLOBAL ROUTINE IMPLIED_DISABLE : NOVALUE =
: 563 0559 1
: 564 0560 1 ++
: 565 0561 1 Functional description:
: 566 0562 1
: 567 0563 1 For all implicitly disabled operators create an operator disable
: 568 0564 1 message using the info in the operator control block, and CALL the
: 569 0565 1 operator enable message handler as if the user had sent the message.
: 570 0566 1 Note that notification of the operator disable is NOT sent to the
: 571 0567 1 operator. This is because the terminal is no longer an operator
: 572 0568 1 terminal, and the user now at the terminal doesn't need to see the
: 573 0569 1 message.
: 574 0570 1
: 575 0571 1 Input:
: 576 0572 1
: 577 0573 1 None.
: 578 0574 1
: 579 0575 1 Implicit Input:
: 580 0576 1
: 581 0577 1 DISABLED_OPER_Q : The list head of all implicitly disabled operators.
: 582 0578 1
: 583 0579 1 Output:
: 584 0580 1
: 585 0581 1 None.
: 586 0582 1
: 587 0583 1 Implicit output:
: 588 0584 1
: 589 0585 1 None.
: 590 0586 1
: 591 0587 1 Side effects:
: 592 0588 1
: 593 0589 1 As operators are disabled, more implicitly disabled operators may
: 594 0590 1 be discovered. If so, they will be inserted on the queue, and
: 595 0591 1 processed in turn. Likewise, as operators are disabled, some requests
: 596 0592 1 may lose operator coverage. These requests will be canceled and
: 597 0593 1 the user notified.
: 598 0594 1
: 599 0595 1 Routine value:
: 600 0596 1
: 601 0597 1 None.
: 602 0598 1 --
: 603 0599 1
: 604 0600 2 BEGIN ! Start of IMPLIED_DISABLE
: 605 0601 2
: 606 0602 2 EXTERNAL
: 607 0603 2 DISABLED_OPER_Q : VECTOR; ! List head of disabled operators
: 608 0604 2
: 609 0605 2 LOCAL
: 610 0606 2 STATUS;
: 611 0607 2
: 612 0608 2 STATUS = 1; ! *** TEMP ***
: 613 0609 2
: 614 0610 1 END; ! End of IMPLIED_DISABLE

```

.EXTRN DISABLED_OPER_Q

50 0000 00000 .ENTRY IMPLIED DISABLE, Save nothing : 0558
01 D0 00002 MOVL #1, STATUS : 0608
04 00005 RET : 0610

: Routine Size: 6 bytes, Routine Base: \$CODE\$ + 0260

: 615 0611 1
: 616 0612 1 END ! End of TIMESTAMP
: 617 0613 0 ELUDOM

PSECT SUMMARY

| Name | Bytes | Attributes |
|-----------|-------|--------------------------------------------------------------|
| \$GLOBALS | 4 | NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2) |
| \$OWNS | 40 | NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2) |
| \$CODE\$ | 614 | NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2) |

Library Statistics

| File | Symbols | | Pages Mapped | Processing Time |
|-----------------------------------------|---------|----------------|--------------|-----------------|
| | Total | Loaded Percent | | |
| _\$255\$DUA28:[SYSLIB]LIB.L32;1 | 18619 | 21 0 | 1000 | 00:01.8 |
| _\$255\$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1 | 633 | 31 4 | 43 | 00:00.9 |

COMMAND QUALIFIERS

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

: 618 0614 0
: Size: 614 code + 44 data bytes
: Run Time: 00:15.3
: Elapsed Time: 00:55.5
: Lines/CPU Min: 2406
: Lexemes/CPU-Min: 16295
: Memory Used: 155 pages
: Compilation Complete

0292 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

This image shows a grid of approximately 20 columns and 15 rows of terminal window displays. Each window contains text-based data, likely representing system logs or diagnostic information. Several windows are highlighted with larger, bold text labels:

- PASDEF LIS** (row 2, column 11)
- PAS102 LIS** (row 3, column 13)
- STATUS LIS** (row 4, column 9)
- LINKDOWN LIS** (row 4, column 11)
- SHUTDOWN LIS** (row 5, column 9)
- SHARE LIS** (row 6, column 7)
- PAS101 LIS** (row 6, column 11)
- PAS103 LIS** (row 7, column 13)
- PASCAL** (row 11, column 11)
- PASLINE LIS** (row 12, column 11)
- TIMESTAMP LIS** (row 14, column 9)

The text within the windows is mostly illegible due to low contrast and small font size, but appears to be structured data or code listings.