


```

RRRRRRRR      EEEEEEEEEE  PPPPPPPP  LL      YY      YY  MM      MM      AAA/AA      IIIIII  NN      NN
RRRRRRRR      EEEEEEEEEE  PPPPPPPP  LL      YY      YY  MM      MM      AAAAAA      IIIIII  NN      NN
RR      RR      EE          PP      PP  LL      YY      YY  MMMM  MMMM  AA      AA      I      NN      NN
RR      RR      EE          PP      PP  LL      YY      YY  MMMM  MMMM  AA      AA      I      NN      NN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AA      AA      I      NNNN  NN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AA      AA      I      NNNN  NN
RRRRRRRR      EEEEEEEEEE  PPPPPPPP  LL      YY      YY  MM      MM  AA      AA      I      NN      NN
RRRRRRRR      EEEEEEEEEE  PPPPPPPP  LL      YY      YY  MM      MM  AA      AA      I      NN      NN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AAAAAAAAAA  I      NN      NNNN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AAAAAAAAAA  I      NN      NNNN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AA      AA      I      NN      NN
RR      RR      EE          PP      PP  LL      YY      YY  MM      MM  AA      AA      I      NN      NN
RR      RR      EEEEEEEEEE  PP      LL      YY      YY  MM      MM  AA      AA      IIIIII  NN      NN
RR      RR      EEEEEEEEEE  PP      LL      YY      YY  MM      MM  AA      AA      IIIIII  NN      NN

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      I      SS
LL      II     SS
LL      III    SS
LL      III    SS
LL      III    SSSSSS
LL      III    SSSSSS
LL      III    SS
LL      III    SS
LL      III    SS
LL      III    SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```

```

1 0001 0 MODULE opcsreplymain ( %TITLE 'REPLY command main module' %SBTTL 'Copyright notice'
2 0002 0
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 IDENT = 'V04-000',
5 0005 0 MAIN = replymain_main
6 0006 0 ) =
7 0007 0
8 0008 0 *****
9 0009 0 *
10 0010 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
11 0011 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
12 0012 0 * ALL RIGHTS RESERVED. *
13 0013 0 *
14 0014 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
15 0015 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
16 0016 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
17 0017 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
18 0018 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
19 0019 0 * TRANSFERRED. *
20 0020 0 *
21 0021 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
22 0022 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
23 0023 0 * CORPORATION. *
24 0024 0 *
25 0025 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
26 0026 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
27 0027 0 *
28 0028 0 *****
29 0029 0
30 0030 0 **
31 0031 0 FACILITY:
32 0032 0
33 0033 0 REPLY command
34 0034 0
35 0035 0 ABSTRACT:
36 0036 0
37 0037 0 This module contains the top level logic for the DCL REPLY command.
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 CW Hobbs
46 0046 0
47 0047 0 Creation date:
48 0048 0
49 0049 0 1-Aug-1983
50 0050 0
51 0051 0 Revision history:
52 0052 0
53 0053 0 V03-003 CWH3169 CW Hobbs 5-May-1984
54 0054 0 Second pass for cluster-wide OPCOM:
55 0055 0 - Check for oper privs, and return NOOPER priv if not there.
56 0056 0 - Add CSID to clm header.
57 0057 0 - Force all batch jobs to /NONOTIFY.

```

OPCSREPLYMAIN
V04-000

REPLY command main module
Copyright notice

J 8
16-Sep-1984 01:44:54
14-Sep-1984 12:50:54

VAX-11 Bliss-32 V4.0-742
[OPCOM.SRC]REPLYMAIN.B32;1

Page 2
(1)

OP
VO

```
: 58      0058 0 !  
: 59      0059 0 !  
: 60      0060 0 !  
: 61      0061 0 !  
: 62      0062 0 !  
: 63      0063 0 !--  
- Return status codes with OPCS_facility set.  
V03-002 CWH3002      CW Hobbs      14-Apr-1984  
Change the two SCS node items to the single SYIS_SCSNODE item
```

```

: 65      0064 1 BEGIN                                %SBTTL 'Start of REPLYMAIN'
: 66      0065 1
: 67      0066 1 LIBRARY 'SYSS:LIBRARY:LIB.L32';
: 68      0067 1 LIBRARY 'LIBS:OPCOMLIB';
: 69      0068 1
: 70      0069 1 FORWARD ROUTINE
: 71      0070 1     replymain_broadcast,           ! Mid-level routine to handle terminal broadcasts
: 72      0071 1     replymain_broadcast_local,     ! Routine to broadcast locally
: 73      0072 1     replymain_fuldev : NOVALUE,     ! Get full device name for terminal, do some checking
: 74      0073 1     replymain_init,                ! Initializations
: 75      0074 1     replymain_logfile,             ! Open or close the log file
: 76      0075 1     replymain_main,                ! Entry point, main routine
: 77      0076 1     replymain_oprenable,           ! Enable or disable operator's terminal
: 78      0077 1     replymain_reply,               ! Reply to a user's request
: 79      0078 1     replymain_status;              ! Give status for a single terminal
: 80      0079 1
: 81      0080 1 EXTERNAL ROUTINE
: 82      0081 1     replybrd_format,                ! Format the reply message
: 83      0082 1     replybrd_io,                    ! Do the actual break through I/O
: 84      0083 1     share_lookup_oper_bit,          ! Convert text string to operator bit number
: 85      0084 1     share_trnlog : NOVALUE;         ! Recursively translate a name
: 86      0085 1
: 87      0086 1 EXTERNAL
: 88      0087 1     reply_image,                     ! Flag, 1 means REPLY image, 0 means OPCOM image
: 89      0088 1     oper_keytbl : VECTOR [, LONG];  ! Keyword table for /ENABLE and /DISABLE qualifiers
: 90      0089 1
: 91      0090 1 OWN
: 92      0091 1     dvi_terminal_len,                ! Length of terminal name
: 93      0092 1     dvi_terminal_buf : VECTOR [max_dev_nam, BYTE],
: 94      0093 1     jpi_username_len,
: 95      0094 1     jpi_username_buf : VECTOR [12, BYTE],
: 96      0095 1     jpi_privs : $bblock [8],
: 97      0096 1     nodename_buf : VECTOR [16, BYTE],
: 98      0097 1     nodename_desc : $stat_str_desc (0, nodename_buf),
: 99      0098 1     tranlog_desc : $stat_str_desc (16, nodename_buf),
100      0099 1     devchar : $bblock [4],
101      0100 1     in_VAXcluster : LONG,
102      0101 1     batch_mode : LONG,
103      0102 1     nodec$id : LONG,
104      0103 1     dvi_items : VECTOR [7, LONG] PRESET (
105      0104 1         [0] = (dvi$devchar^16 OR 4),
106      0105 1         [1] = devchar,
107      0106 1         [2] = 0,
108      0107 1         [3] = (dvi$fulldevnam^16 OR max_dev_nam),
109      0108 1         [4] = dvi_terminal_buf,
110      0109 1         [5] = dvi_terminal_len,
111      0110 1         [6] = 0),
112      0111 1     mba2_refcnt : LONG,
113      0112 1     mba2_dvi_items : VECTOR [4, LONG] PRESET (
114      0113 1         [0] = (dvi$refcnt^16 OR 4),
115      0114 1         [1] = mba2_refcnt,
116      0115 1         [2] = 0,
117      0116 1         [3] = 0),
118      0117 1     jpi_items : VECTOR [7, LONG] PRESET (
119      0118 1         [0] = (jpi$username^16 OR 12),
120      0119 1         [1] = jpi_username_buf,
121      0120 1         [2] = jpi_username_len,

```

```

122      0121 1          [3] = (jpi$ curpriv^16 OR 8),
123      0122 1          [4] = jpi$ privs,
124      0123 1          [5] = 0,
125      0124 1          [6] = 0),
126      0125 1          syi_items : VECTOR [10, LONG] PRESET (
127      0126 1          [0] = (syi$ nodename^16 OR 16),
128      0127 1          [1] = n$dename_buf,
129      0128 1          [2] = nodename_desc [dsc$w length],
130      0129 1          [3] = (syi$ node_csid^16 OR 4),
131      0130 1          [4] = nodecsid,
132      0131 1          [5] = 0,
133      0132 1          [6] = (syi$ cluster_member^16 OR 4),
134      0133 1          [7] = in_VAXcluster,
135      0134 1          [8] = 0,
136      0135 1          [9] = 0);
137
138      0137 1          :: Define ascii text descriptors once
139
140      0139 1          BIND
141      0140 1          ascid_ABORT      = %ASCID 'ABORT',
142      0141 1          ascid_ALL        = %ASCID 'ALL',
143      0142 1          ascid_BELL       = %ASCID 'BELL',
144      0143 1          ascid_BLANK_TAPE = %ASCID 'BLANK TAPE',
145      0144 1          ascid_DISABLE   = %ASCID 'DISABLE',
146      0145 1          ascid_ENABLE     = %ASCID 'ENABLE',
147      0146 1          ascid_INITIALIZE_TAPE = %ASCID 'INITIALIZE_TAPE',
148      0147 1          ascid_LOG        = %ASCID 'LOG',
149      0148 1          ascid_MBA2       = %ASCID 'MBA2:',
150      0149 1          ascid_NODE       = %ASCID 'NODE',
151      0150 1          ascid_NOTIFY     = %ASCID 'NOTIFY',
152      0151 1          ascid_P1        = %ASCID 'P1',
153      0152 1          ascid_PENDING    = %ASCID 'PENDING',
154      0153 1          ascid_SHUTDOWN   = %ASCID 'SHUTDOWN',
155      0154 1          ascid_STATUS     = %ASCID 'STATUS',
156      0155 1          ascid_SYSCOMMAND = %ASCID 'SYSSCOMMAND',
157      0156 1          ascid_SYSNODE    = %ASCID 'SYSSNODE',
158      0157 1          ascid_TEMPORARY = %ASCID 'TEMPORARY',
159      0158 1          ascid_TERMINAL  = %ASCID 'TERMINAL',
160      0159 1          ascid_TO         = %ASCID 'TO',
161      0160 1          ascid_URGENT     = %ASCID 'URGENT',
162      0161 1          ascid_USERNAME   = %ASCID 'USERNAME',
163      0162 1          ascid_WAIT       = %ASCID 'WAIT';

```

```

165 0163 1 GLOBAL ROUTINE replymain_broadcast = %SBTTL 'replymain_broadcast routine'
166 0164 1 ++
167 0165 1 Functional description:
168 0166 1
169 0167 1 This routine controls terminal broadcasts.
170 0168 1
171 0169 1 Input:
172 0170 1
173 0171 1 None.
174 0172 1
175 0173 1 Implicit Input:
176 0174 1
177 0175 1 None.
178 0176 1
179 0177 1 Output:
180 0178 1
181 0179 1 None.
182 0180 1
183 0181 1 Implicit output:
184 0182 1
185 0183 1 None.
186 0184 1
187 0185 1 Side effects:
188 0186 1
189 0187 1 None.
190 0188 1
191 0189 1 Routine value:
192 0190 1
193 0191 1 None.
194 0192 1 --
195 0193 1
196 0194 2 BEGIN ! Start of replymain_broadcast
197 0195 2
198 0196 2 OWN
199 0197 2 node_csid : LONG,
200 0198 2 targnode_itmlst : VECTOR [4, LONG] PRESET (
201 0199 2 [0] = (syi$_node_csid^16 OR 4),
202 0200 2 [1] = node_csid,
203 0201 2 [2] = 0,
204 0202 2 [3] = 0);
205 0203 2
206 0204 2 REGISTER
207 0205 2 mlen, ! Output message length
208 0206 2 mptr : $ref_bvector; ! Output message pointer
209 0207 2
210 0208 2 LOCAL
211 0209 2 text : $dyn_str_desc, ! Dynamic string descr for message text
212 0210 2 message : $bblock [opc$_maxread], ! Buffer to build message
213 0211 2 message_desc : VECTOR [2, LONG],
214 0212 2 status;
215 0213 2
216 0214 2 ! Check for oper priv, return nooper error with opcom's facility code
217 0215 2
218 0216 2 IF NOT .jpi_privs [prv$_oper]
219 0217 2 THEN
220 0218 2 RETURN (opc$_facility^16 OR ss$_nooper);
221 0219 2

```

```
222 0220 2 | Initialize the message
223 0221 2 |
224 0222 2 | NOTE: We are using an internal interface to OPCOM which is subject to change!
225 0223 2 |
226 0224 2 CH$FILL (0, rpybrd_k_min_size, message);           ! Init all fixed fields to zero
227 0225 2 message [clm_b_rqstcode] = opc$x_clusmsg;
228 0226 2 message [clm_b_clm_code] = clm_rpybrd_local;
229 0227 2 message [clm_b_ds_version] = rpybrd_k_ds_version;
230 0228 2 message [clm_b_sw_version] = opc$k_sw_version;
231 0229 2 message [clm_l_csid] = .nodecsid;
232 0230 2 |
233 0231 2 | Find out which qualifiers are present, and set the bits in the message
234 0232 2 |
235 0233 2 message [rpybrd_v_all] = cli$present (ascid_ALL);
236 0234 2 message [rpybrd_v_bell] = cli$present (ascid_BELL);
237 0235 2 message [rpybrd_v_node] = cli$present (ascid_NODE);
238 0236 2 message [rpybrd_v_notify] = cli$present (ascid_NOTIFY);
239 0237 2 message [rpybrd_v_shutdown] = cli$present (ascid_SHUTDOWN);
240 0238 2 message [rpybrd_v_terminal] = cli$present (ascid_TERMINAL);
241 0239 2 message [rpybrd_v_urgent] = cli$present (ascid_URGENT);
242 0240 2 message [rpybrd_v_username] = cli$present (ascid_USERNAME);
243 0241 2 message [rpybrd_v_wait] = cli$present (ascid_WAIT);
244 0242 2 IF .batch_mode ! Make adjustment for batch mode
245 0243 2 THEN
246 0244 2 message [rpybrd_v_notify] = false; ! /NOTIFY doesn't make much sense for batch
247 0245 2 |
248 0246 2 | Move the standard fields to the message, first the sending terminal name
249 0247 2 |
250 0248 2 mptr = message [rpybrd_t_text]; ! Set output pointer to start of text area
251 0249 2 mlen = .dvi_terminal_len; ! Get length of terminal name
252 0250 2 message [rpybrd_w_send_term_len] = .mlen; ! Store the length in the message header
253 0251 2 CH$MOVE (.mlen, dvi_terminal_buf, .mptr); ! Append the name to the buffer
254 0252 2 mptr = .mptr + .mlen; ! Move the output pointer past this item
255 0253 2 |
256 0254 2 | Next move the username of the sender
257 0255 2 |
258 0256 2 mlen = .jpi_username_len;
259 0257 2 message [rpybrd_w_send_user_len] = .mlen;
260 0258 2 CH$MOVE (.mlen, jpi_username_buf, .mptr);
261 0259 2 mptr = .mptr + .mlen;
262 0260 2 |
263 0261 2 | Move the csid and nodename of the sender
264 0262 2 |
265 0263 2 message [rpybrd_l_send_csid] = .nodecsid;
266 0264 2 mlen = .nodename_desc [dsc$w_length];
267 0265 2 message [rpybrd_w_send_node_len] = .mlen;
268 0266 2 CH$MOVE (.mlen, .nodename_desc [dsc$a_pointer], .mptr);
269 0267 2 mptr = .mptr + .mlen;
270 0268 2 |
271 0269 2 | Next fetch and move the actual message text
272 0270 2 |
273 0271 2 IF NOT (status = cli$get_value (ascid_P1, text))
274 0272 2 THEN
275 0273 2 $signal_stop (.status);
276 0274 2 mlen = .text [dsc$w_length]; ! Zero-length messages are fine with us
277 0275 2 message [rpybrd_w_message_len] = .mlen;
278 0276 2 CH$MOVE (.mlen, .text [dsc$a_pointer], .mptr);
```



```

279 0277 2 mptr = .mptr + .mlen;
280 0278 2 message [rpybrd_w_optional_off] = .mptr - message;      ! Save offset to start of optional items
281 0279 2
282 0280 2 : Now we move the optional items. These come from the /TERMINAL=(...), /USERNAME=(...) and /NODE=(...).
283 0281 2 : These items are stored as counted ASCII (ASCIC) items. The length stored in the fixed length field
284 0282 2 : describes the total length of all of the ASCIC items.
285 0283 2
286 0284 2 : Move the targeted terminals to the message
287 0285 2
288 0286 2 IF .message [rpybrd_v_terminal]
289 0287 2 THEN
290 0288 2     WHILE cli$get_value (ascid_TERMINAL, text)
291 0289 2     DO
292 0290 2         BEGIN
293 0291 2         share_trnlog (text);                                ! Translate the name
294 0292 2         replymain_fuldev (text);                            ! Convert to full device name and check
295 0293 2         mlen = .text [dsc$w_length];                          ! Get the length of this item
296 0294 2         mptr [0] = .mlen;                                     ! Put the length in the message (ASCIC)
297 0295 2         CH$MOVE (.mlen, .text [dsc$a_pointer], mptr [1]);    ! Put the data after the length
298 0296 2         mlen = .mlen + 1;                                     ! Adjust the length for the count byte
299 0297 2         message [rpybrd_w_targ_term_len] =                    ! Add this item to the total length
300 0298 2             .message [rpybrd_w_targ_term_len] + .mlen;
301 0299 2         mptr = .mptr + .mlen;
302 0300 2     END;
303 0301 2
304 0302 2 : Move targeted usernames
305 0303 2
306 0304 2 IF .message [rpybrd_v_username]
307 0305 2 THEN
308 0306 2     WHILE cli$get_value (ascid_USERNAME, text)
309 0307 2     DO
310 0308 2         BEGIN
311 0309 2         share_trnlog (text);                                ! Translate the name
312 0310 2         mlen = .text [dsc$w_length];                          ! Get the length of this item
313 0311 2         mptr [0] = .mlen;                                     ! Put the length in the message (ASCIC)
314 0312 2         CH$MOVE (.mlen, .text [dsc$a_pointer], mptr [1]);    ! Put the data after the length
315 0313 2         mlen = .mlen + 1;                                     ! Adjust the length for the count byte
316 0314 2         message [rpybrd_w_targ_user_len] =                    ! Add this item to the total length
317 0315 2             .message [rpybrd_w_targ_user_len] + .mlen;
318 0316 2         mptr = .mptr + .mlen;
319 0317 2     END;
320 0318 2
321 0319 2 : Now try the target node names
322 0320 2
323 0321 2 message [rpybrd_v_broad_local] = true;                       ! Assume it is going to the local node
324 0322 2 message [rpybrd_v_broad_remoteall] = true;                  ! Assume it is going to all remote nodes
325 0323 2 IF .message [rpybrd_v_node]
326 0324 2 THEN
327 0325 2     BEGIN
328 0326 2     :
329 0327 2     : If /NODE is present, assume it is not going to the local node until we know more. We know
330 0328 2     : that it will not be going to all nodes (case where every node is in the list is not interesting).
331 0329 2     :
332 0330 2     message [rpybrd_v_broad_local] = false;
333 0331 2     message [rpybrd_v_broad_remoteall] = false;
334 0332 2     message [rpybrd_w_targ_node_off] = .mptr - message; ! Save offset to node area for CLUSREPLY local code
335 0333 2

```

```
336 0334 3 ! Get each of the nodenames from the command line, and add them to the message buffer
337 0335 3
338 0336 3
339 0337 3
340 0338 4
341 0339 4
342 0340 4
343 0341 4
344 0342 4
345 0343 4
346 0344 4
347 0345 4
348 0346 4
349 0347 4
350 0348 4
351 0349 4
352 0350 4
353 0351 4
354 0352 5
355 0353 5
356 0354 5
357 0355 5
358 0356 5
359 0357 5
360 0358 5
361 0359 5
362 0360 5
363 0361 5
364 0362 5
365 0363 5
366 0364 6
367 0365 6
368 0366 6
369 0367 5
370 0368 5
371 0369 5
372 0370 5
373 0371 5
374 0372 5
375 0373 5
376 0374 5
377 0375 6
378 0376 5
379 0377 5
380 0378 5
381 0379 5
382 0380 5
383 0381 5
384 0382 5
385 0383 5
386 0384 4
387 0385 3
388 0386 3
389 0387 3
390 0388 3
391 0389 3
392 0390 3

! Get each of the nodenames from the command line, and add them to the message buffer
WHILE cli$get_value (ascid_NODE, text)
DO
  BEGIN
  share_trnlog (text); ! Translate the name
  ! If the translated name is the same as the local name, set the local node flag.
  IF CH$EQL (.text [dsc$w_length], .text [dsc$a_pointer],
    .nodename_desc [dsc$w_length], .nodename_desc [dsc$a_pointer], 0)
  THEN
    message [rpybrd_v_broad_local] = true ! Now we know it is going to the local node
  ! If the node is not the same, verify that it is in the cluster. Then place the csid of the node
  ! in the message.
  ELSE
  BEGIN
  LOCAL
  ptr,
  desc : VECTOR [2, LONG];
  ! Remove leading "_" and trailing ":" from the node name, $GETSYI doesn't like them
  desc [0] = .text [dsc$w_length]; ! Make a local fixed descriptor for the dynamic string
  desc [1] = .text [dsc$a_pointer];
  ptr = CH$FIND_NOT_CH (.desc [0], .desc [1], %C '_');
  IF .ptr NEQ 0
  THEN
  BEGIN
  desc [0] = .desc [0] - (.ptr - .desc [1]);
  desc [1] = .ptr;
  END;
  ptr = CH$FIND_CH (.desc [0], .desc [1], %C ':');
  IF .ptr NEQ 0
  THEN
  desc [0] = .ptr - .desc [1];
  ! Do a $GETSYI to get information about the target system (the csid)
  IF NOT (status = $getsyi (nodename=desc, itmlst=targnode_itmlst))
  THEN
  $signal_stop (opc$_valuerr, 1, text, .status);
  mlen = 5; ! Get the length of this item (plus count by
  mptr [0] = .mlen; ! Put the length in the message (ASCII)
  (mptr [1]) = .node_csid; ! Put the data after the length
  message [rpybrd_w_targ_node_len] = ! Add this item to the total length
    .message [rpybrd_w_targ_node_len] + .mlen;
  mptr = .mptr + .mlen;
  END;
  END;
  ! If there were no node names, then the local node is the only node to get the message
  IF .message [rpybrd_w_targ_node_len] EQL 0
  THEN
```

```

393 0391 3      message [rpybrd_v_broad_local] = true      ! Now we know it is going to the local node
394 0392 3      ELSE
395 0393 3      message [rpybrd_v_broad_remotelst] = true;    ! This means that nodes in a list
396 0394 2      END;
397 0395 2      :
398 0396 2      Almost done with the message, store the final length in the header and build a descriptor
399 0397 2      :
400 0398 2      message_desc [0] = message [clm_w_length] = .mptr - message;    ! Save in header and descriptor
401 0399 2      message_desc [1] = message;
402 0400 2      :
403 0401 2      Now, decide if we should let the OPCOM process do the actual i/o or whether we should do it locally.
404 0402 2      We do it locally if any of the following conditions are true:
405 0403 2      :
406 0404 2      - If the command is REPLY /WAIT, then the user has specifically requested local i/o operations.
407 0405 2      - If the reference count on the operator mailbox is not equal to 2, then OPCOM is not there, and
408 0406 2      we have to do it. (Also if the $getdvi fails, MBA2: is not there. Should not be possible)
409 0407 2      - If the $sndopr fails, then obviously OPCOM won't do it and we must.
410 0408 2      :
411 0409 2      IF .message [rpybrd_v_wait]
412 0410 2      THEN
413 0411 2      RETURN replymain_broadcast_local (message);
414 0412 2      :
415 0413 2      Check the operator mailbox
416 0414 2      :
417 0415 2      status = $getdvi (devnam=ascid_MBA2, itmlst=mba2_dvi_items);
418 0416 2      IF NOT .status
419 0417 2      OR
420 0418 2      .mba2_refcnt NEQ 2
421 0419 2      THEN
422 0420 2      RETURN replymain_broadcast_local (message);
423 0421 2      :
424 0422 2      Send the message to OPCOM so that it will get to remote nodes
425 0423 2      :
426 0424 3      IF NOT (status = $sndopr (msgbuf=message_desc))
427 0425 2      THEN
428 0426 2      RETURN replymain_broadcast_local (message);
429 0427 2      :
430 0428 2      RETURN (opc$_facility^16 OR ss$_normal);
431 0429 1      END;

```

! End of replymain_broadcast

.TITLE OPC\$REPLYMAIN REPLY command main module
.IDENT \V04-000\

.PSECT \$SPLITS,NOWRT,NOEXE,2

```

00 00 00 54 52 4F 42 41 00000 P.AAB: .ASCII \ABORT\<0><0><0>
010E0005 00008 P.AAA: .LONG 17694725
00000000' 0000C .ADDRESS P.AAB
00 4C 4C 41 00010 P.AAD: .ASCII \ALL\<0>
010E0003 00014 P.AAC: .LONG 17694723
00000000' 00018 .ADDRESS P.AAD
4C 4C 45 42 0001C P.AAF: .ASCII \BELL\
010E0004 00020 P.AAE: .LONG 17694724
00000000' 00024 .ADDRESS P.AAF
00 00 45 50 41 54 5F 4B 4E 41 4C 42 00028 P.AAH: .ASCII \BLANK TAPE\<0><0>
010E000A 00034 P.AAG: .LONG 17694730

```

```

00 45 4C 42 41 53 49 44 00038 .ADDRESS P.AAH
00 45 4C 42 41 53 49 44 0003C P.AAJ: .ASCII \DISABLE\<0>
010E0007 00044 P.AAI: .LONG 17694727
00000000' 00048 .ADDRESS P.AAJ
00 00 45 4C 42 41 4E 45 0004C P.AAL: .ASCII \ENABLE\<0><0>
010E0006 00054 P.AAK: .LONG 17694726
00000000' 00058 .ADDRESS P.AAL
45 50 41 54 5F 45 5A 49 4C 41 49 54 49 4E 49 0005C P.AAN: .ASCII \INITIALIZE_TAPE\<0>
00 00 00 00 00 00 00 00 0006B
010E000F 0006C P.AAM: .LONG 17694735
00000000' 00070 .ADDRESS P.AAM
00 47 4F 4C 00074 P.AAP: .ASCII \LOG\<0>
010E0003 00078 P.AAO: .LONG 17694723
00000000' 0007C .ADDRESS P.AAP
00 00 3A 32 41 42 4D 5F 00080 P.AAR: .ASCII \MBA2:\<0><0>
010E0006 00088 P.AAQ: .LONG 17694726
00000000' 0008C .ADDRESS P.AAR
45 44 4F 4E 00090 P.AAT: .ASCII \NODE\
010E0004 00094 P.AAS: .LONG 17694724
00000000' 00098 .ADDRESS P.AAT
00 00 59 46 49 54 4F 4E 0009C P.AAV: .ASCII \NOTIFY\<0><0>
010E0006 000A4 P.AAU: .LONG 17694726
00000000' 000A8 .ADDRESS P.AAV
00 00 31 50 000AC P.AAX: .ASCII \P1\<0><0>
010E0002 000B0 P.AAW: .LONG 17694722
00000000' 000B4 .ADDRESS P.AAX
00 47 4E 49 44 4E 45 50 000B8 P.AAZ: .ASCII \PENDING\<0>
010E0007 000C0 P.AAY: .LONG 17694727
00000000' 000C4 .ADDRESS P.AAZ
4E 57 4F 44 54 55 48 53 000C8 P.ABB: .ASCII \SHUTDOWN\
010E0008 000D0 P.ABA: .LONG 17694728
00000000' 000D4 .ADDRESS P.ABB
00 00 53 55 54 41 54 53 000D8 P.ABD: .ASCII \STATUS\<0><0>
010E0006 000E0 P.ABC: .LONG 17694726
00000000' 000E4 .ADDRESS P.ABD
00 44 4E 41 4D 4D 4F 43 24 53 59 53 000E8 P.ABF: .ASCII \SYS$COMMAND\<0>
010E000B 000F4 P.ABE: .LONG 17694731
00000000' 000F8 .ADDRESS P.ABF
45 44 4F 4E 24 53 59 53 000FC P.ABH: .ASCII \SYS$NODE\
010E0008 00104 P.ABG: .LONG 17694728
00000000' 00108 .ADDRESS P.ABH
00 00 00 59 52 41 52 4F 50 4D 45 54 0010C P.ABJ: .ASCII \TEMPORARY\<0><0><0>
010E0009 00118 P.ABI: .LONG 17694729
00000000' 0011C .ADDRESS P.ABJ
4C 41 4E 49 4D 52 45 54 00120 P.ABL: .ASCII \,ERMINAL\
010E0008 00128 P.ABK: .LONG 17694728
00000000' 0012C .ADDRESS P.ABL
00 00 4F 54 00130 P.ABN: .ASCII \TO\<0><0>
010E0002 00134 P.ABM: .LONG 17694722
00000000' 00138 .ADDRESS P.ABN
00 00 54 4E 45 47 52 55 0013C P.ABP: .ASCII \URGENT\<0><0>
010E0006 00144 P.ABO: .LONG 17694726
00000000' 00148 .ADDRESS P.ABP
45 4D 41 4E 52 45 53 55 0014C P.ABR: .ASCII \USERNAME\
010E0008 00154 P.ABQ: .LONG 17694728
00000000' 00158 .ADDRESS P.ABR
54 49 41 57 0015C P.ABT: .ASCII \WAIT\

```

.....

.....

```
010E0004 00160 P.ABS: .LONG 17694724
00000000' 00164 .ADDRESS P.ABT
                .PSECT $OWNS,NOEXE,2
00000 DVI_TERMINAL_LEN:
                .BLKB 4
00004 DVI_TERMINAL_BUF:
                .BLKB 64
00044 JPI_USERNAME_LEN:
                .BLKB 4
00048 JPI_USERNAME_BUF:
                .BLKB 12
00054 JPI_PRIVS:
                .BLKB 8
0005C NODENAME_BUF:
                .BLKB 16
0000 0006C NODENAME_DESC:
                .WORD 0
01 0E 0006E .BYTE 14, 1
00000000' 00070 .ADDRESS NODENAME_BUF
0010 00074 TRANLOG_DESC:
                .WORD 16
01 0E 00076 .BYTE 14, 1
00000000' 00078 .ADDRESS NODENAME_BUF
0007C DEVCHAR: .BLKB 4
00080 IN_VAXCLUSTER:
                .BLKB 4
00084 BATCH_MODE:
                .BLKB 4
00088 NODECSID:
                .BLKB 4
00020004 0008C DVI_ITEMS:
                .LONG 131076
00000000' 00090 .ADDRESS DEVCHAR
00E80040 00000000' 00094 .LONG 0, 15204416
00000000' 0009C .ADDRESS DVI_TERMINAL_BUF, DVI_TERMINAL_LEN
00000000' 000A4 .LONG 0
000A8 MBA2_REFCNT:
                .BLKB 4
001E0004 000AC MBA2_DVI_ITEMS:
                .LONG 1966084
00000000' 000B0 .ADDRESS MBA2_REFCNT
00000000' 000B4 .LONG 0, 0
0202000C 000BC JPI_ITEMS:
                .LONG 33685516
00000000' 000C0 .ADDRESS JPI_USERNAME_BUF, JPI_USERNAME_LEN
04000008 000C8 .LONG 67108872
00000000' 000CC .ADDRESS JPI_PRIVS
00000000' 000D0 .LONG 0, 0
10D90010 000D8 SYI_ITEMS:
                .LONG 282656784
00000000' 000DC .ADDRESS NODENAME_BUF, NODENAME_DESC
10DU0004 000E4 .LONG 282066948
00000000' 000E8 .ADDRESS NODECSID
10CF0004 000EC .LONG 0, 282001412
00000000' 000F4 .ADDRESS IN_VAXCLUSTER
```

```

00000000 00000000 000F8 .LONG 0, 0
00100 NODE_CSID:
          10D00004 00104 TARGNODE .BLKB 4
          .ITMLST:
00000000' 00108 .LONG 282066948
00000000 000C0000 0010C .ADDRESS NODE_CSID
          .LONG 0, 0

```

```

ASCID_ABORT= P.AAA
ASCID_ALL= P.AAC
ASCID_BELL= P.AAE
ASCID_BLANK_TAPE= P.AAG
ASCID_DISABLE= P.AAI
ASCID_ENABLE= P.AAK
ASCID_INITIALIZE_TAPE=
P.AAM
ASCID_LOG= P.AAO
ASCID_MBA2= P.AAQ
ASCID_NODE= P.AAS
ASCID_NOTIFY= P.AAU
ASCID_P1= P.AAW
ASCID_PENDING= P.AAY
ASCID_SHUTDOWN= P.ABA
ASCID_STATUS= P.ABC
ASCID_SYSCOMMAND= P.ABE
ASCID_SYSNODE= P.ABG
ASCID_TEMPORARY= P.ABI
ASCID_TERMINAL= P.ABK
ASCID_TO= P.ABM
ASCID_URGENT= P.ABO
ASCID_USERNAME= P.ABQ
ASCID_WAIT= P.ABS

```

```

.EXTRN REPLYBRD_FORMAT
.EXTRN REPLYBRD_IO, SHARE_LOOKUP_OPER_BIT
.EXTRN SHARE_TRNLOG, REPLY_IMAGE
.EXTRN OPER_KEYTBL, CLISPRESENT
.EXTRN CLISGET_VALUE, LIB$STOP
.EXTRN SYSSGETSYI, SYSSGETDVI
.EXTRN SYSSNDOPR

```

```
.PSECT $CCDE$,NOWRT,2
```

```

          OFFC 00000 .ENTRY REPLYMAIN_BROADCAST, Save R2,R3,R4,R5,R6,- ; 0163
          R7,R8,R9,R10,R11
          5B 0000' CF 9E 00002 MOVAB ASCID_NODE, R11
          5A 0000' CF 9E 00007 MOVAB NODECSID, R10
          59 00000000G 00 9E 0000C MOVAB CLISPRESENT, R9
          5E F5E8 CE 9E 00013 MOVAB -2584(SP), SP
F8 AD 020E0000 8F D0 00018 MOVL #34471936, TEXT ; 0209
          FC AD D4 00020 CLRL TEXT+4
          08 CE AA 02 E0 00023 BBS #2, JPI_PRIVS+2, 1$ ; 0216
          50 00052894 8F D0 00028 MOVL #338068, R0 ; 0218
          04 0002F RET
          00 6E 00 2C 00030 1$: MOVCS #0, (SP), #0, #48, MESSAGE ; 0224
          10 AE 09061013 8F D0 00037 MOVL #151392275, MESSAGE ; 0225
          18 AE 6A D0 0003F MOVL NODECSID, MESSAGE+8 ; 0229

```

				80	AB 9F 00043	PUSHAB	ASCID ALL		0233
		69			01 FB 00046	CALLS	#1, C[IS]PRESENT		
1C	AE	01	00		50 FO 00049	INSV	RO, #0, #1, MESSAGE+12		
		69		8C	AB 9F 0004F	PUSHAB	ASCID BELL		0234
		01	01		01 FB 00052	CALLS	#1, C[IS]PRESENT		
1C	AE	01	01		50 FO 00055	INSV	RO, #1, #1, MESSAGE+12		
		69			5B DD 00058	PUSHL	R11		0235
		01	02		01 FB 0005D	CALLS	#1, C[IS]PRESENT		
1C	AE	01	02		50 FO 00060	INSV	RO, #2, #1, MESSAGE+12		
		69		10	AB 9F 00066	PUSHAB	ASCID NOTIFY		0236
		01	03		01 FB 00069	CALLS	#1, C[IS]PRESENT		
1C	AE	01	03		50 FO 0006C	INSV	RO, #3, #1, MESSAGE+12		
		69		3C	AB 9F 00072	PUSHAB	ASCID SHUTDOWN		0237
		01	04		01 FB 00075	CALLS	#1, C[IS]PRESENT		
1C	AE	01	04		50 FO 00078	INSV	RO, #4, #1, MESSAGE+12		
		69		0094	CB 9F 0007E	PUSHAB	ASCID TERMINAL		0238
		01	05		01 FB 00082	CALLS	#1, C[IS]PRESENT		
1C	AE	01	05		50 FO 00085	INSV	RO, #5, #1, MESSAGE+12		
		69		00B0	CB 9F 0008B	PUSHAB	ASCID URGENT		0239
		01	06		01 FB 0008F	CALLS	#1, C[IS]PRESENT		
1C	AE	01	06		50 FO 00092	INSV	RO, #6, #1, MESSAGE+12		
		69		00C0	CB 9F 00098	PUSHAB	ASCID USERNAME		0240
		01	07		01 FB 0009C	CALLS	#1, C[IS]PRESENT		
1C	AE	01	07		50 FO 0009F	INSV	RO, #7, #1, MESSAGE+12		
		69		00CC	CB 9F 000A5	PUSHAB	ASCID WAIT		0241
		01	00		01 FB 000A9	CALLS	#1, C[IS]PRESENT		
1D	AE	01	00		50 FO 000AC	INSV	RO, #0, #1, MESSAGE+13		
		04		FC	AA E9 000B2	BLBC	BATCH MODE, 2\$		0242
	1C	AE	57		08 8A 000B6	BICB2	#8, MESSAGE+12		0244
		56		40	AE 9E 000BA	MOVAB	MESSAGE+48, MPTR		0248
		67	FF7C	FF78	CA D0 000BE	MOVL	DVI TERMINAL_LEN, MLEN		0249
		56			56 B0 000C3	MOVW	MLEN, MESSAGE+20		0250
		24	AE		56 28 000C7	MOVW	MLEN, MESSAGE+20		0251
		67	FF7C		56 C0 000CD	ADDL2	MLEN, MPTR		0252
		56		BC	AA D0 000D0	MOVL	JPI USERNAME_LEN, MLEN		0256
		26	AE		56 B0 000D4	MOVW	MLEN, MESSAGE+22		0257
		67	CO		56 28 000D8	MOVW	MLEN, JPI USERNAME_BUF, (MPTR)		0258
		57			56 C0 000DD	ADDL2	MLEN, MPTR		0259
		20	AE		6A D0 000E0	MOVL	NODECSID, MESSAGE+16		0263
		56		E4	AA 3C 000E4	MOVZWL	NODENAME_DESC, MLEN		0264
		28	AE		56 B0 000E8	MOVW	MLEN, MESSAGE+24		0265
		67	E8		56 29 000EC	MOVW	MLEN, MESSAGE+24		0266
		57			56 C0 000F1	ADDL2	MLEN, @NODENAME_DESC+4, (MPTR)		0267
				F8	AD 9F 000F4	PUSHAB	TEXT		0271
				1C	AB 9F 000F7	PUSHAB	ASCID P1		
		00000000G	00		02 FB 000FA	CALLS	#2, C[IS]GET_VALUE		
			58		50 D0 00101	MOVL	RO, STATUS		
			0A		58 E8 00104	BLBS	STATUS, 3\$		
		00000000G	00		58 DD 00107	PUSHL	STATUS		0273
					01 FB 00109	CALLS	#1, LIB\$STOP		
			56	F8	AD 3C 00111	RET			0274
			AE		56 B0 00115	MOVZWL	TEXT, MLEN		0275
		67	2A FC		56 28 00119	MOVW	MLEN, MESSAGE+26		0276
			BD		56 C0 0011E	MOVW	MLEN, @TEXT+4, (MPTR)		0277
			57		56 C0 0011E	ADDL2	MLEN, MPTR		0277
			50	10	AE 9E 00121	MOVAB	MESSAGE, RO		0278
2C	AE		57		50 A3 00125	SUBW3	RO, MPTR, MESSAGE+28		

39	1C	AE	05	E1	0012A	BBC	#5, MESSAGE+12, 5\$	0286
			F8	AD	9F 0012F	PUSHAB	TEXT	0288
	00000000G	00	0094	CB	9F 00132	PUSHAB	ASCID, TERMINAL	
		28		02	FB 00136	CALLS	#2, C[ISGET_VALUE	
				50	E9 0013D	BLBC	RO, 5\$	
	0000G	CF	F8	AD	9F 00140	PUSHAB	TEXT	0291
				01	FB 00143	CALLS	#1, SHARE_TRNLOG	
	0000V	CF	F8	AD	9F 00148	PUSHAB	TEXT	0292
		56		01	FB 0014B	CALLS	#1, REPLYMAIN_FULDEV	
		67	F8	AD	3C 00150	MOVZWL	TEXT, MLEN	0293
01	A7	FC		56	90 00154	MOVB	MLEN, (MPTR)	0294
		BD		56	28 00157	MOV3	MLEN, @TEXT+4, 1(MPTR)	0295
				56	D6 0015D	INCL	MLEN	0296
	2E	AE		56	A0 0015F	ADDW2	MLEN, MESSAGE+30	0298
		57		56	C0 00163	ADDL2	MLEN, MPTR	0299
				C7	11 00166	BRB	4\$	0288
			1C	AE	95 00168	TSTB	MESSAGE+12	0304
				31	18 0016B	BGEQ	7\$	
			F8	AD	9F 0016D	PUSHAB	TEXT	0306
	00000000G	00	00C0	CB	9F 00170	PUSHAB	ASCID, USERNAME	
		20		02	FB 00174	CALLS	#2, C[ISGET_VALUE	
				50	E9 0017B	BLBC	RO, 7\$	
	0000G	CF	F8	AD	9F 0017E	PUSHAB	TEXT	0309
		56		01	FB 00181	CALLS	#1, SHARE TRNLOG	
		67	F8	AD	3C 00186	MOVZWL	TEXT, MLEN	0310
01	A7	FC		56	90 0018A	MOVB	MLEN, (MPTR)	0311
		BD		56	28 0018D	MOV3	MLEN, @TEXT+4, 1(MPTR)	0312
				56	D6 00193	INCL	MLEN	0313
	30	AE		56	A0 00195	ADDW2	MLEN, MESSAGE+32	0315
		57		56	C0 00199	ADDL2	MLEN, MPTR	0316
				CF	11 0019C	BRB	6\$	0306
	1D	AE		06	88 0019E	BISB2	#6, MESSAGE+13	0322
03	1C	AE		02	E0 001A2	BBS	#2, MESSAGE+12, 8\$	0323
				00BF	31 001A7	BRW	20\$	
	1D	AE		06	8A 001AA	BICB2	#6, MESSAGE+13	0331
		50	10	AE	9E 001AE	MOVAB	MESSAGE, RO	0332
34	AE	57		50	A3 001B2	SUBW3	RO, MPTR, MESSAGE+36	
			F8	AD	9F 001B7	PUSHAB	TEXT	0336
	00000000G	00		5B	DD 001BA	PUSHL	R11	
		03		02	FB 001BC	CALLS	#2, CLISGET_VALUE	
				50	E8 001C3	BLBS	RO, 10\$	
				0091	31 001C6	BRW	18\$	
			F8	AD	9F 001C9	PUSHAB	TEXT	0339
	0000G	CF		01	FB 001CC	CALLS	#1, SHARE TRNLOG	
E4	AA	00		F8	AD 2D 001D1	CMPC5	TEXT, @TEXT+4, #0, NODENAME_DESC, -	0343
		BD		EB	BA 001D9		@NODENAME_DESC+4	
				06	12 001DB	BNEQ	12\$	
	1D	AE		02	88 001DD	BISB2	#2, MESSAGE+13	0346
				D4	11 001E1	BRB	9\$	
	6E		F8	AD	3C 001E3	MOVZWL	TEXT, DESC	0359
	04	AE		FC	AD 00 001E7	MOVL	TEXT+4, DESC+4	0360
04	BE	6E		SF	8F 3B 001EC	SKPC	#95, DESC, @DESC+4	0361
				02	12 001F2	BNEQ	13\$	
				51	D4 001F4	CLRL	R1	
				51	D5 001F6	TSTL	PTR	0362
				0C	13 001F8	BEQL	14\$	
	50	04		51	C3 001FA	SUBL3	PTR, DESC+4, RO	0365

04	BE	04	6E AE 6E	50	CO	001FF	ADDL2	R0, DESC	0366
				51	DO	00202	MOVL	PTR, DESC+4	0368
				3A	3A	00206	LOCC	#58, DESC, @DESC+4	
				02	12	0020B	BNEQ	15\$	
				51	D4	0020D	CLRL	R1	
				51	D5	0020F	TSTL	PTR	0369
				05	13	00211	BEQL	16\$	
	6E		51	04	AE	C3 00213	SUBL3	DESC+4, PTR, DESC	0371
					7E	7C 00218	CLRQ	-(SP)	0375
					7E	D4 0021A	CLRL	-(SP)	
				7C	AA	9F 0021C	PUSHAB	TARGNODE_ITMLST	
				10	AE	9F 0021F	PUSHAB	DESC	
					7E	7C 00222	CLRQ	-(SP)	
	00000000G		00		07	FB 00224	CALLS	#7, SYSSGETSYI	
			58		50	DO 0022B	MOVL	R0, STATUS	
			15		58	E8 0022E	BLBS	STATUS, 17\$	
					58	DD 00231	PUSHL	STATUS	0377
				F8	AD	9F 00233	PUSHAB	TEXT	
					01	DD 00236	PUSHL	#1	
	00000000G		00		8F	DD 00238	PUSHL	#361052	
				0005825C	04	FB 0023E	CALLS	#4, LIB\$STOP	
					04	00245	RET		
			56		05	DO 00246	MOVL	#5, MLEN	0378
			67		56	90 00249	MOVAB	MLEN, (MPTR)	0379
			01		AA	DO 0024C	MOVL	NODE_CSID, 1(MPTR)	0380
			32		56	A0 00251	ADDW2	MLEN, MESSAGE+34	0382
			57		56	CO 00255	ADDL2	MLEN, MPTR	0383
					87	11 00258	BRB	11\$	0336
					AE	B5 0025A	TSTW	MESSAGE+34	0389
				32	06	12 0025D	BNEQ	19\$	
			1D		02	88 0025F	BISB2	#2, MESSAGE+13	0391
					04	11 00263	BRB	20\$	
			1D		08	88 00265	BISB2	#8, MESSAGE+13	0393
			50		AE	9E 00269	MOVAB	MESSAGE, R0	0398
			57		50	C2 0026D	SUBL2	R0, R7	
			14		57	B0 00270	MOVW	R7, MESSAGE+4	
			08		57	DO 00274	MOVL	R7, MESSAGE_DESC	
			OC		AE	9E 00278	MOVAB	MESSAGE, MESSAGE_DESC+4	0399
			31		AE	E8 0027D	BLBS	MESSAGE+13, 21\$	0409
				1D	7E	7C 00281	CLRQ	-(SP)	0415
					7E	7C 00283	CLRQ	-(SP)	
				24	AA	9F 00285	PUSHAB	MBA2_DVI_ITEMS	
				F4	AB	9F 00288	PUSHAB	ASCID_MBA2	
					7E	7C 0028B	CLRQ	-(SP)	
	00000000G		00		08	FB 0028D	CALLS	#8, SYSSGETDVI	
			58		50	DO 00294	MOVL	R0, STATUS	
			18		58	E9 00297	BLBC	STATUS, 21\$	0416
			02		20	AA	D1 0029A	CPL	MBA2_REFcnt, #2
					12	12 0029E	BNEQ	21\$	0418
					7E	D4 002A0	CLRL	-(SP)	0424
				OC	AE	9F 002A2	PUSHAB	MESSAGE_DESC	
	000C0000G		00		02	FB 002A5	CALLS	#2, SYSSNDOPR	
			58		50	DO 002AC	MOVL	R0, STATUS	
			09		58	E8 002AF	BLBS	STATUS, 22\$	
					AE	9F 002B2	PUSHAB	MESSAGE	0426
	0000V		CF		10	01	FB 002B5	CALLS	#1, REPLYMAIN_BROADCAST_LOCAL
					04	002BA	RET		


```

433 0430 1 GLOBAL ROUTINE replymain_broadcast_local (message : $ref_bblock) = %SBTTL 'replymain_broadcast_local (m
434 0431 1  **
435 0432 1  Functional description:
436 0433 1
437 0434 1      This routine broadcasts to terminals on the local node.
438 0435 1
439 0436 1  Input:
440 0437 1
441 0438 1      message - pointer to RPYBRD message
442 0439 1
443 0440 1  Implicit Input:
444 0441 1
445 0442 1      None.
446 0443 1
447 0444 1  Output:
448 0445 1
449 0446 1      None.
450 0447 1
451 0448 1  Implicit output:
452 0449 1
453 0450 1      None.
454 0451 1
455 0452 1  Side effects:
456 0453 1
457 0454 1      None.
458 0455 1
459 0456 1  Routine value:
460 0457 1
461 0458 1      I/O status
462 0459 1  --
463 0460 1
464 0461 2 BEGIN                                ! Start of replymain_broadcast_local
465 0462 2
466 0463 2 LOCAL
467 0464 2     status;
468 0465 2
469 0466 2 :
470 0467 2 : If we thought we were going to talk to the cluster, let them know it ain't a gonna happen.
471 0468 2
472 0469 2 IF .in_VAXcluster
473 0470 2     AND
474 0471 2     (.message [rpybrd_v_broad_remoteall]
475 0472 2     OR
476 0473 2     .message [rpybrd_v_broad_remotelst])
477 0474 2 THEN
478 0475 2     $signal (IF .message [rpybrd_v_wait] THEN opc$_noremwait ELSE opc$_norembroad);
479 0476 2 :
480 0477 2 : Format and broadcast the message to the local node
481 0478 2
482 0479 2 message [rpybrd_v_wait] = true;                                ! We are in /WAIT mode now, perhaps implicitly
483 0480 2 IF .message [rpybrd_v_broad_local]
484 0481 2 THEN
485 0482 2     BEGIN
486 0483 2     status = replybrd_format (.message, nodename_desc);
487 0484 2     IF .status
488 0485 2     THEN
489 0486 2     status = replybrd_io (.message, nodename_desc);

```

```

: 490      0487 3      END
: 491      0488 2      ELSE
: 492      0489 2      status = opc$_nolclbroad;
: 493      0490 2
: 494      0491 2      RETURN (opc$_facility^16 OR .status);
: 495      0492 1      END;

```

! End of replymain_broadcast_local

			0004	00000	.ENTRY	REPLYMAIN BROADCAST_LOCAL, Save R2	: 0430
		29	0000'	CF E9 00002	BLBC	IN VAXCLUSTER, 4\$: 0469
		5C	04	AC D0 00007	MOVL	MESSAGE, R0	: 0471
05	OD	A0		02 E0 0000B	BBS	#2, 13(R0), 1\$	
1D	OD	A0		03 E1 00010	BBC	#3, 13(R0), 4\$: 0473
		50	04	AC D0 00015	MOVL	MESSAGE, R0	: 0475
		08	0D	A0 E9 00019	BLBC	13(R0), 2\$	
			000582D0	8F DD 0001D	PUSHL	#361168	
				06 11 00023	BRB	3\$	
			000582C8	8F DD 00025	PUSHL	#361160	
	00000000G	00		01 FB 0002B	CALLS	#1, LIB\$SIGNAL	
		52	04	AC D0 00032	MOVL	MESSAGE, R2	: 0479
		OD		01 88 00036	BISB2	#1, 13(R2)	
1B	OD	A2		01 E1 0003A	BBC	#1, 13(R2), 5\$: 0480
			0000'	CF 9F 0003F	PUSHAB	NODENAME_DESC	: 0483
				52 DD 00043	PUSHL	R2	
	0000G	CF		02 FB 00045	CALLS	#2, REPL BRD_FORMAT	
		14		50 E9 0004A	BLBC	STATUS, 6\$: 0484
			0000'	CF 9F 0004D	PUSHAB	NODENAME_DESC	: 0486
				52 DD 00051	PUSHL	R2	
	0000G	CF		02 FB 00053	CALLS	#2, REPLYBRD_10	
				07 11 00058	BRB	6\$: 0480
		50	000582C0	8F D0 0005A	MOVL	#361152, STATUS	: 0489
		50	00050000	8F C8 00061	BISL2	#327680, R0	: 0491
				04 00068	RET		: 0492

; Routine Size: 105 bytes, Routine Base: \$CODE\$ + 02C3

```

497 0493 1 GLOBAL ROUTINE replymain_fuldev (name : $ref_bblock) : NOVALUE = %SBTTL 'replymain_fuldev (name : $re
498 0494 1
499 0495 1 !++
500 0496 1 Functional description:
501 0497 1
502 0498 1 Convert terminal name to full (SCS) device name. Make sure that a device name which fails contains
503 0499 1 a valid SCS nodename for a node in our cluster, plus at least three more letters (e.g. DELPHISTT0)
504 0500 1
505 0501 1 Input:
506 0502 1
507 0503 1 name - Address of dynamic string descriptor for input name
508 0504 1
509 0505 1 Implicit Input:
510 0506 1
511 0507 1 None.
512 0508 1
513 0509 1 Output:
514 0510 1
515 0511 1 name - Receives a new dynamic string if we find the device on our system
516 0512 1
517 0513 1 Implicit output:
518 0514 1
519 0515 1 None.
520 0516 1
521 0517 1 Side effects:
522 0518 1
523 0519 1 None.
524 0520 1
525 0521 1 Routine value:
526 0522 1
527 0523 1 None.
528 0524 1 --
529 0525 1
530 0526 2 BEGIN ! Start of replymain_fuldev
531 0527 2
532 0528 2 LOCAL
533 0529 2 len,
534 0530 2 ptr,
535 0531 2 p,
536 0532 2 desc : VECTOR [2, LONG],
537 0533 2 status;
538 0534 2
539 0535 2
540 0536 2 If the input string is not dynamic, scream and shout.
541 0537 2
542 0538 2 IF .name [dsc$b_class] NEQ dsc$k_class_d
543 0539 2 THEN
544 0540 2 $signal_stop (ss$_badparam);
545 0541 2
546 0542 2 See if we can get a local device name from the input
547 0543 2
548 0544 2 IF (status = $getdvi (devnam=.name, itmlst=dvi_items))
549 0545 2 THEN
550 0546 2 BEGIN
551 0547 2
552 0548 2 Copy the dvi string to the output
553 0549 2

```

```

554 0550 3 desc [0] = .dvi_terminal_len;
555 0551 3 desc [1] = dvi_terminal_buf;
556 0552 4 IF NOT (status = str$copy_dx (.name, desc))
557 0553 3 THEN
558 0554 3 $signal_stop (.status);
559 0555 3 RETURN;
560 0556 2 END;
561 0557 2
562 0558 2 : If we are not in a VAXcluster, nothing more to do with the name. It is wrong.
563 0559 2
564 0560 2 IF NOT .in_VAXcluster
565 0561 2 THEN
566 0562 2 $signal_stop (.status);
567 0563 2
568 0564 2 : Not a local device, make sure it looks somewhat like a valid remote device. For the sake of argument,
569 0565 2 imagine that a valid remote device name looks like 'nnnnn$xxx' where 'nnnnn' is a node which is
570 0566 2 actually in our cluster and 'xxx' is a least three letters (can any valid terminal be shorter than T10?)
571 0567 2
572 0568 2 len = .name [dsc$w_length];
573 0569 2 ptr = .name [dsc$a_pointer];
574 0570 2 p = CH$FIND_CH (.len, .ptr, %C '$'); ! Find the dollar sign
575 0571 2
576 0572 2 : If there is no dollar sign, or if there are fewer than three letters after the '$', or the '$' is the
577 0573 2 first letter then there is no such device.
578 0574 2
579 0575 2 IF .p EQL 0
580 0576 2 OR
581 0577 2 .p EQL .ptr
582 0578 2 THEN
583 0579 2 $signal_stop (opc$_val_err, 1, .name, ss$_nosuchdev);
584 0580 2 IF .len - (.p - .ptr) LSS 3
585 0581 2 THEN
586 0582 2 $signal_stop (opc$_valuerr, 1, .name, ss$_nosuchdev);
587 0583 2
588 0584 2 : Found something that could be a node name, remove the '$xxx' from the string
589 0585 2
590 0586 2 len = .p - .ptr;
591 0587 2
592 0588 2 : If any leading underscores, skip over them
593 0589 2
594 0590 2 p = CH$FIND_NOT_CH (.len, .ptr, %C '_');
595 0591 2 IF .p NEQ 0
596 0592 2 THEN
597 0593 2 BEGIN
598 0594 2 len = .len - (.p - .ptr);
599 0595 2 ptr = .p;
600 0596 2 END;
601 0597 2 IF .len LSS 0
602 0598 2 THEN
603 0599 2 $signal_stop (opc$_valuerr, 1, .name, ss$_nosuchdev);
604 0600 2
605 0601 2 : Ok, we should have a good node name, try it out by doing a $GETSYI on the node (any info will do)
606 0602 2
607 0603 2 desc [0] = .len;
608 0604 2 desc [1] = .ptr;
609 0605 2 IF NOT (status = $getsyi (nodename=desc, itm1st=syi_items))
610 0606 2 THEN

```

```

: 611 0607 2  $signal_stop (opc$_valuerr, 1, .name, .status);
: 612 0608 2  :
: 613 0609 2  | We've got something that looks like a good name, but of course it could be _DELPHISDUA169:. We seem to
: 614 0610 2  | have two choices. One is to make some assumptions about what a terminal name looks like, the other
: 615 0611 2  | would be to actually talk to the other node and see if it has the device. It isn't a good idea to
: 616 0612 2  | assume anything about a device name (boy have we learned that lesson!), and it seems to be pretty
: 617 0613 2  | expensive to have a chat with the other node. Actually, we have a third choice, which is to leave
: 618 0614 2  | things as they stand. If the guy really wants to know if he succeeded, he will use /NOTIFY.
: 619 0615 2  |
: 620 0616 2  RETURN;
: 621 0617 1  END;

```

! End of replymain_fuldev

				.EXTRN STR\$COPY_DX					
				.ENTRY	REPLYMAIN_FULDEV, Save R2,R3,R4,R5,R6,R7	:			
	57	00000000G	00	9E	00002	MOVAB	LIB\$STOP, R7	: 0493	
	5E		08	C2	00009	SUBL2	#8, SP		
	53	04	AC	DO	0000C	MOVL	NAME, R3	: 0538	
	02	03	A3	91	00010	CMPB	3(R3), #2		
			04	13	00014	BEQL	1\$		
			14	DD	00016	PUSHL	#20	: 0540	
			3D	11	00018	BRB	4\$		
			7E	7C	0001A	1\$: CLRQ	-(SP)	: 0544	
			7E	7C	0001C	CLRQ	-(SP)		
		0000'	CF	9F	0001E	PUSHAB	DVI_ITEMS		
			53	DD	00022	PUSHL	R3		
			7E	7C	00024	CLRQ	-(SP)		
	00000000G	00	08	FB	00026	CALLS	#8, SYSS\$GETDVI		
		56	50	DO	0002D	MOVL	R0, STATUS		
		1D	56	E9	00030	BLBC	STATUS, 2\$		
	04	6E	0000'	CF	DO	00033	MOVL	DVI_TERMINAL_LEN, DESC	: 0550
		AE	0000'	CF	9E	00038	MOVAB	DVI_TERMINAL_BUF, DESC+4	: 0551
			4008	8F	BB	0003E	PUSHR	#^M<R3, SP>	: 0552
	00000000G	00	02	FB	00042	CALLS	#2, STR\$COPY_DX		
		56	50	DO	00049	MOVL	R0, STATUS		
		06	56	E9	0004C	BLBC	STATUS, 3\$		
			04	0004F	RET			: 0554	
	06	0000'	CF	E8	00050	2\$: BLBS	IN VAXCLUSTER, 5\$: 0560	
			56	DD	00055	3\$: PUSHL	STATUS	: 0562	
			67	01	FB	00057	4\$: CALLS	#1, LIB\$STOP	
				04	0005A	RET			
	54		63	3C	0005B	5\$: MOVZWL	(R3), LEN	: 0568	
	52	04	A3	DO	0005E	MOVL	4(R3), PTR	: 0569	
	62		24	3A	00062	LOCC	#36, LEN, (PTR)	: 0570	
			02	12	00066	BNEQ	6\$		
			51	D4	00068	CLRL	R1		
			55	51	DO	0006A	6\$: MOVL	R1, P	
			31	13	0006D	BEQL	9\$: 0575	
			52	55	D1	0006F	CPL	P, PTR	: 0577
			2C	13	00072	BEQL	9\$		
	50		52	C3	00074	SUBL3	PTR, P, R0	: 0580	
			51	A0	9E	00078	MOVAB	2(R0), R1	
			51	D1	0007C	CPL	LEN, R1		
			1F	19	0007F	BLSS	9\$		
			54	50	DO	00081	MOVL	R0, LEN	: 0586

62	54	5F	8F	3B	00084	S&PC	#95, LEN, (PTR)	: 0590
			02	12	00089	BNEQ	7\$:
			51	D4	0008B	CLRL	R1	:
	55		51	D0	0008D	7\$:	MOVL R1, P	:
			0A	13	0009C	BEQL	8\$: 0591
50	52		55	C3	00092	SUBL3	P, PTR, R0	: 0594
	54		50	C0	00096	ADDL2	R0, LEN	:
	52		55	D0	00099	MOVL	P, PTR	: 0595
			54	D5	0009C	8\$:	TSTL LEN	: 0597
			07	18	0009E	BGEQ	10\$:
	7E	0908	8F	3C	000A0	9\$:	MOVZWL #2312, -(SP)	: 0599
			23	11	000A5	BRB	11\$:
	6E		54	D0	000A7	10\$:	MOVL LEN, DESC	: 0603
04	AE		52	D0	0C0AA	MOVL	PTR, DESC+4	: 0604
			7E	7C	000AE	CLRQ	-(SP)	: 0605
			7E	D4	000B0	CLRL	-(SP)	:
		0000'	CF	9F	000B2	PUSHAB	SYI ITEMS	:
		10	AE	9F	000B6	PUSHAB	DESC	:
			7E	7C	000B9	CLRQ	-(SP)	:
00000000G	00		07	FB	000BB	CALLS	#7, SYSSGETSYI	:
	56		50	D0	000C2	MOVL	R0, STATUS	:
	0F		56	E8	000C5	BLBS	STATUS, 12\$:
			56	DD	000C8	PUSHL	STATUS	: 0607
			53	DD	000CA	11\$:	PUSHL R3	:
			01	DD	000CC	PUSHL	#1	:
		0005825C	8F	DD	000CE	PUSHL	#361052	:
	67		04	FB	000D4	CALLS	#4, LIB\$STOP	:
			04	00	000D7	12\$:	RET	: 0617

: Routine Size: 216 bytes, Routine Base: \$CODE\$ + 032C


```

: 623 0618 1 GLOBAL ROUTINE replymain_init =          %SBTTL 'replymain_init routine'
: 624 0619 1
: 625 0620 1 |++
: 626 0621 1 | Functional description:
: 627 0622 1 |
: 628 0623 1 |         This is the initialization routine for REPLY.  Various common initializations are done.
: 629 0624 1 |
: 630 0625 1 | Input:
: 631 0626 1 |
: 632 0627 1 |         None.
: 633 0628 1 |
: 634 0629 1 | Implicit Input:
: 635 0630 1 |
: 636 0631 1 |         None.
: 637 0632 1 |
: 638 0633 1 | Output:
: 639 0634 1 |
: 640 0635 1 |         None.
: 641 0636 1 |
: 642 0637 1 | Implicit output:
: 643 0638 1 |
: 644 0639 1 |         None.
: 645 0640 1 |
: 646 0641 1 | Side effects:
: 647 0642 1 |
: 648 0643 1 |         None.
: 649 0644 1 |
: 650 0645 1 | Routine value:
: 651 0646 1 |
: 652 0647 1 |         None.
: 653 0648 1 | --
: 654 0649 1
: 655 0650 2 BEGIN                                ! Start of replymain_init
: 656 0651 2
: 657 0652 2 LOCAL
: 658 0653 2     ptr : $ref_bblock,
: 659 0654 2     status;
: 660 0655 2
: 661 0656 2 | Some routines which are shared with OPCOM need to know whether REPLY is running or OPCOM is running.
: 662 0657 2 | Let them know.
: 663 0658 2
: 664 0659 2 reply_image = 1;
: 665 0660 2
: 666 0661 2 | Do a $GETJPI to get information about the current process
: 667 0662 2
: 668 0663 3 IF NOT (status = $getjpi (itmlst=jpi_items))
: 669 0664 2 THEN
: 670 0665 2     $signal_stop (.status);
: 671 0666 2
: 672 0667 2 | Get the actual length of the username, since it is blank padded to 12 bytes
: 673 0668 2
: 674 0669 2 ptr = CH$FIND_CH (12, jpi_username_buf, %C ' ');
: 675 0670 2 IF .ptr NEQ 0
: 676 0671 2 THEN
: 677 0672 2     jpi_username_len = .ptr - jpi_username_buf;
: 678 0673 2
: 679 0674 2 | Do a $GETSYI to get information about the current system

```

```

: 680      0675      2  |
: 681      0676      3  | IF NOT (status = $getsyi (itmlst=ysi_items))
: 682      0677      2  | THEN
: 683      0678      2  |     $signal_stop (.status);
: 684      0679      2  |
: 685      0680      2  |     Get the length of the node name, since it is blank padded to 8 bytes
: 686      0681      2  |
: 687      0682      2  | ptr = CH$FIND_CH (8, nodename_buf, %C ' ');
: 688      0683      2  | IF .ptr NEQ 0
: 689      0684      2  | THEN
: 690      0685      2  |     nodename_desc [dsc$w_length] = .ptr - nodename_buf;
: 691      0686      2  |
: 692      0687      2  |     If the SCS nodename is null, try to translate SYS$NODE to find the DECnet name. Remove the '_' and ':'
: 693      0688      2  |     from the translated name.
: 694      0689      2  |
: 695      0690      2  | IF .nodename_desc [dsc$w_length] EQL 0
: 696      0691      2  | THEN
: 697      0692      3  |     BEGIN
: 698      0693      4  |     IF NOT (status = $trnlog (lognam=ascid_SYSNODE, rslten=tranlog_desc, rslbuf=tranlog_desc, dsbmsk=6))
: 699      0694      3  |     THEN
: 700      0695      3  |         $signal_stop (.status);
: 701      0696      3  |     IF .status EQL ss$_normal
: 702      0697      3  |     THEN
: 703      0698      4  |         BEGIN
: 704      0699      4  |         ptr = CH$FIND_NOT_CH (.tranlog_desc [dsc$w_length], .tranlog_desc [dsc$a_pointer], %C '_');
: 705      0700      4  |         IF .ptr NEQ 0
: 706      0701      4  |         THEN
: 707      0702      5  |             BEGIN
: 708      0703      5  |             tranlog_desc [dsc$w_length] = .tranlog_desc [dsc$w_length] - (.ptr - .tranlog_desc [dsc$a_pointe
: 709      0704      5  |             tranlog_desc [dsc$a_pointer] = .ptr;
: 710      0705      4  |             END;
: 711      0706      4  |         ptr = CH$FIND_CH (.tranlog_desc [dsc$w_length], .tranlog_desc [dsc$a_pointer], %C ':');
: 712      0707      4  |         IF .ptr NEQ 0
: 713      0708      4  |         THEN
: 714      0709      4  |             tranlog_desc [dsc$w_length] = .ptr - .tranlog_desc [dsc$a_pointer];
: 715      0710      4  |             nodename_desc [dsc$w_length] = .tranlog_desc [dsc$w_length];
: 716      0711      4  |             nodename_desc [dsc$a_pointer] = .tranlog_desc [dsc$a_pointer];
: 717      0712      3  |         END;
: 718      0713      2  |     END;
: 719      0714      2  |
: 720      0715      2  |     Do a $GETDVI to get the name of the command terminal.
: 721      0716      2  |
: 722      0717      3  | IF NOT (status = $ygetdvi (devnam=ascid_SYSCOMMAND, itmlst=dvi_items))
: 723      0718      2  | THEN
: 724      0719      2  |     $signal_stop (.status);
: 725      0720      2  | IF NOT .devchar [dev$v_trm]
: 726      0721      2  | THEN
: 727      0722      3  |     BEGIN
: 728      0723      3  |     dvi_terminal_len = 6 + .nodename_desc [dsc$w_length];
: 729      0724      3  |     CH$COPY (.nodename_desc [dsc$w_length], .nodename_desc [dsc$a_pointer],
: 730      0725      3  |             6, UPLIT BYTE (' Batch'), 0, .dvi_terminal_len, dvi_terminal_buf);
: 731      0726      3  |     batch_mode = true;
: 732      0727      2  |     END;
: 733      0728      2  |
: 734      0729      2  | RETURN .status;
: 735      0730      1  | END;

```

! End of replymain_init

```

.PSECT $SPLITS$,NOWRT,NOEXE,2
      68 63 74 61 42 20 00168 P.ABU: .ASCII \ Batch\
.PSECT $CODES$,NOWRT,2
      07FC 00000 .ENTRY REPLYMAIN_INIT, Save R2,R3,R4,R5,R6,R7,R8,- ; 0618
      00C0G 5A 0000' CF 9E 00002 MOVAB TRANLOG_DESC, R10
      CF 01 D0 00007 MOVL #1, REPLY_IMAGE ; 0659
      7E 7C 0000C CLRQ -(SP) ; 0663
      7E D4 0000E CLRL -(SP)
      48 AA 9F 00010 PUSHAB JPI_ITEMS
      7E 7C 00013 CLRQ -(SP)
      7E D4 00015 CLRL -(SP)
      00000000G 00 07 FB 00017 CALLS #7, SYSS$GETJPI
      59 50 D0 0001E MOVL R0, STATUS
      61 59 E9 00021 BLBC STATUS, 5$
      D4 AA 0C 20 3A 00024 LOCC #32, #12, JPI_USERNAME_BUF ; 0669
      02 12 00029 BNEQ 1$
      51 D4 0002B CLRL R1
      52 51 D0 0002D 1$: MOVL R1, PTR
      09 13 00030 BEQL 2$ ; 0670
      50 52 D4 AA 9E 00032 MOVAB JPI_USERNAME_BUF, R0 ; 0672
      D0 AA 52 50 C3 00036 SUBW3 R0, PTR, JPI_USERNAME_LEN
      7E 7C 0003B 2$: CLRQ -(SP) ; 0676
      7E D4 0003D CLRL -(SP)
      64 AA 9F 0003F PUSHAB SYI_ITEMS
      7E 7C 00042 CLRQ -(SP)
      7E D4 00044 CLRL -(SP)
      00000000G 00 07 FB 00046 CALLS #7, SYSS$GETSYI
      59 50 D0 0004D MOVL R0, STATUS
      32 59 E9 00050 BLBC STATUS, 5$
      E8 AA 08 20 3A 00053 LOCC #32, #8, NODENAME_BUF ; 0682
      02 12 0005B BNEQ 3$
      51 D4 0005A CLRL R1
      52 51 D0 0005C 3$: MOVL R1, PTR
      09 13 0005F BEQL 4$ ; 0683
      50 52 E8 AA 9E 00061 MOVAB NODENAME_BUF, R0 ; 0685
      F8 AA 52 50 A3 00065 SUBW3 R0, PTR, NODENAME_DESC
      F8 AA B5 0006A 4$: TSTW NODENAME_DESC ; 0690
      55 12 0006D BNEQ 10$
      06 DD 0006F PUSHL #6 ; 0693
      7E 7C 00071 CLRQ -(SP)
      5A DD 00073 PUSHL R10
      5A DD 00075 PUSHL R10
      0000' CF 9F 00077 PUSHAB ASCID_SYSNODE
      00000000G 00 06 FB 0007B CALLS #6, SYSS$TRNLOG
      59 50 D0 00082 MOVL R0, STATUS
      56 59 E9 00085 5$: BLBC STATUS, 11$
      01 59 D1 00088 CMPL STATUS, #1 ; 0696
      37 12 0008B BNEQ 10$
      04 BA 6A 5F 8F 3B 0008D SKPC #95, TRANLOG_DESC, @TRANLOG_DESC+4 ; 0699

```

				02	12	00093			BNEQ	6\$			
				51	D4	00095			CLRL	R1			
			52	51	D0	00097	6\$:		MOVL	R1, PTR			
				0C	13	0009A			BEQL	7\$			0700
	50	04	AA	52	C3	0009C			SUBL3	PTR, TRANLOG_DESC+4, R0			0703
			6A	50	A0	000A1			ADDW2	R0, TRANLOG_DESC			
				52	D0	000A4			MOVL	PTR, TRANLOG_DESC+4			0704
04	BA		6A	3A	3A	000A8	7\$:		LOCC	#58, TRANLOG_DESC, @TRANLOG_DESC+4			0706
				02	12	000AD			BNEQ	8\$			
				51	D4	000AF			CLRL	R1			
			52	51	D0	000B1	8\$:		MOVL	R1, PTR			
				05	13	000B4			BEQL	9\$			0707
	6A		52	04	AA	A3	000B6		SUBW3	TRANLOG_DESC+4, PTR, TRANLOG_DESC			0709
		F8	AA	04	6A	B0	000BB	9\$:	MOVW	TRANLOG_DESC, NODENAME_DESC			0710
		FC	AA	04	AA	D0	000BF		MOVL	TRANLOG_DESC+4, NODENAME_DESC+4			0711
					7E	7C	000C4	10\$:	CLRQ	-(SP)			0717
					7E	7C	000C6		CLRQ	-(SP)			
				18	AA	9F	000C8		PUSHAB	DVI_ITEMS			
				0000'	CF	9F	000CB		PUSHAB	ASCID_SYSCOMMAND			
					7E	7C	000CF		CLRQ	-(SP)			
	00000000G		00	08	FB	000D1			CALLS	#8, SYSSGETDVI			
			59	50	D0	000D8			MOVL	R0, STATUS			
			0A	59	E8	000DB			BLBS	STATUS, 12\$			
	00000000G		00	59	DD	000DE	11\$:		PUSHL	STATUS			0719
				01	FB	000E0			CALLS	#1, LIB\$STOP			
					04	000E7			RET				
	30	08	AA	02	E0	000E8	12\$:		BBS	#2, DEVCHAR, 14\$			0720
		8C	AA	F8	AA	3C	000ED		MOVZWL	NODENAME_DESC, DVI_TERMINAL_LEN			0723
		8C	AA	06	C0	000F2			ADDL2	#6, DVI_TERMINAL_LEN			
			58	F8	AA	3C	000F6		MOVZWL	NODENAME_DESC, R8			0724
			57	8C	AA	D0	000FA		MOVL	DVI_TERMINAL_LEN, R7			0725
			56	90	AA	9E	000FE		MOVAB	DVI_TERMINAL_BUF, R6			0724
57	00	FC	BA	58	2C	00102			MOVCS	R8, @NODENAME_DESC+4, #0, R7, (R6)			
				66		00108							
				0E	18	00109			BGEQ	13\$			
			56	58	C0	0010B			ADDL2	R8, R6			
			57	58	C2	0010E			SUBL2	R8, R7			
57	00	0000'	CF	06	2C	00111			MOVCS	#6, P.ABU, #0, R7, (R6)			
				66		00118							
			10	01	D0	00119	13\$:		MOVL	#1, BATCH_MODE			0726
			50	59	D0	0011D	14\$:		MOVL	STATUS, R0			0729
				04	04	00120			RET				0730

; Routine Size: 289 bytes, Routine Base: \$CODE\$ + 0404

```

737 0731 1 GLOBAL ROUTINE replymain_logfile = %SBTTL 'replymain_logfile'
738 0732 1 ++
739 0733 1 Functional description:
740 0734 1
741 0735 1 This routine controls closing and opening the operator's log file
742 0736 1
743 0737 1 Input:
744 0738 1
745 0739 1 None.
746 0740 1
747 0741 1 Implicit Input:
748 0742 1
749 0743 1 CLI parameters
750 0744 1
751 0745 1 Output:
752 0746 1
753 0747 1 None.
754 0748 1
755 0749 1 Implicit output:
756 0750 1
757 0751 1 None.
758 0752 1
759 0753 1 Side effects:
760 0754 1
761 0755 1 None.
762 0756 1
763 0757 1 Routine value:
764 0758 1
765 0759 1 None.
766 0760 1 --
767 0761 1
768 0762 2 BEGIN ! Start of replymain_logfile
769 0763 2
770 0764 2 REGISTER
771 0765 2 mlen, ! Output message length
772 0766 2 mptr : $ref_bvector; ! Output message pointer
773 0767 2
774 0768 2 LOCAL
775 0769 2 message : $bblock [128], ! Buffer to build message
776 0770 2 message_desc : VECTOR [2, LONG] PRESET ([1] = message),
777 0771 2 status;
778 0772 2
779 0773 2 Initialize the message
780 0774 2
781 0775 2 NOTE: We are using an internal interface to OPCOM which is subject to change!
782 0776 2
783 0777 2 CH$FILL (0, opc$k_logfile_min_size, message); ! Init all fixed fields to zero
784 0778 2 message [opc$b_rqstcode] = opc$x_logfile;
785 0779 2 message [opc$b_scope] = opc$k_system;
786 0780 2 IF cli$present (ascid_LOG)
787 0781 2 THEN
788 0782 2 $bblock [message [opc$l_rq_options], opc$v_initlog] = true
789 0783 2 ELSE
790 0784 2 $bblock [message [opc$l_rq_options], opc$v_closelog] = true;
791 0785 2
792 0786 2 Move the sending terminal name
793 0787 2

```

```

: 794      0788 2 mptr = message [opc$t_logfile_opr];           ! Set output pointer to start of text area
: 795      0789 ~ mlen = dvi_terminal_len;                       ! Get length of terminal name
: 796      0790 ~ mptr [0] = .mlen;                               ! Store the ASCII length
: 797      0791 ~ CH$MOVE (.mlen, dvi_terminal_buf, mptr [1]); ! Append the name to the buffer
: 798      0792 ~ message_desc [0] = $byteoffset (opc$t_logfile_opr) + 1 + .mlen; ! Save total length
: 799      0793 ~
: 800      0794 ~ Send the message to OPCOM
: 801      0795 ~
: 802      0796 ~ IF NOT (status = $sndopr (msgbuf=message_desc))
: 803      0797 ~ THEN
: 804      0798 ~     $signal_stop (.status);
: 805      0799 ~
: 806      0800 ~ RETURN ss$_normal;
: 807      0801 1 END;                                           ! End of replymain_logfile

```

					007C 0000	.ENTRY	REPLYMAIN_LOGFILE, Save R2,R3,R4,R5,R6	: 0731
		5E	FF7C	CE	9E 00002	MOVAB	-132(SP), -SP	
				7E	D4 00007	CLRL	MESSAGE_DESC	: 0770
		04	08	AE	9E 00009	MOVAB	MESSAGE, MESSAGE_DESC+4	
1E	00	6E		00	2C 0000E	MOVCS	#0, (SP), #0, #30, MESSAGE	: 0777
			08	AE	00013			
		08	AE	010B	8F B0 00015	MOVW	#267, MESSAGE	: 0778
				0000'	CF 9F 0001B	PUSHAB	ASCID LOG	: 0780
		00000000G	00	01	FB 0001F	CALLS	#1, C[ISPRESENT	
			06	50	E9 00026	BLBC	R0, 1\$	
		0E	AE	01	88 00029	BISB2	#1, MESSAGE+6	: 0782
				04	11 0002D	BRB	2\$	
		0E	AE	02	88 0002F 1\$:	BISB2	#2, MESSAGE+6	: 0784
			50	22	AE 9E 00033 2\$:	MOVAB	MESSAGE+26, MPTR	: 0788
			56	0000'	CF D0 00037	MOVL	DVI_TERMINAL_LEN, MLEN	: 0789
			60	56	90 0003C	MOVB	MLEN, (MPTR)	: 0790
	01	A0	0000'	CF	56 28 0003F	MOVCS	MLEN, DVI_TERMINAL_BUF, 1(MPTR)	: 0791
			6E	1B	A6 9E 00046	MOVAB	27(R6), MESSAGE_DESC	: 0792
				7E	D4 0004A	CLRL	-(SP)	: 0796
			04	AE	9F 0004C	PUSHAB	MESSAGE_DESC	
		00000000G	00	02	FB 0004F	CALLS	#2, SYS\$SNDOPR	
			0A	50	E8 00056	BLBS	STATUS, 3\$	
		00000000G	00	50	DD 00059	PUSHL	STATUS	: 0798
				01	FB 0005B	CALLS	#1, LIB\$STOP	
			50	04	00062	RET		
				01	D0 00063 3\$:	MOVL	#1, R0	: 0800
				04	00066	RET		: 0801

; Routine Size: 103 bytes, Routine Base: \$CODES + 0525

```

: 809      0802 1 GLOBAL ROUTINE replymain_main =          %SBTTL 'replymain_main routine'
: 810      0803 1
: 811      0804 1 |++
: 812      0805 1 | Functional description:
: 813      0806 1 |
: 814      0807 1 |     This is the main routine for REPLY.  When REPLY is started, control is transfered here.
: 815      0808 1 |
: 816      0809 1 | Input:
: 817      0810 1 |
: 818      0811 1 |     None.
: 819      0812 1 |
: 820      0813 1 | Implicit Input:
: 821      0814 1 |
: 822      0815 1 |     None.
: 823      0816 1 |
: 824      0817 1 | Output:
: 825      0818 1 |
: 826      0819 1 |     None.
: 827      0820 1 |
: 828      0821 1 | Implicit output:
: 829      0822 1 |
: 830      0823 1 |     None.
: 831      0824 1 |
: 832      0825 1 | Side effects:
: 833      0826 1 |
: 834      0827 1 |     None.
: 835      0828 1 |
: 836      0829 1 | Routine value:
: 837      0830 1 |
: 838      0831 1 |     None.
: 839      0832 1 | --
: 840      0833 1
: 841      0834 2 BEGIN                                  ! Start of replymain_main
: 842      0835 2
: 843      0836 2 LOCAL
: 844      0837 2     status;
: 845      0838 2
: 846      0839 2 |
: 847      0840 2 | Perform common initializations
: 848      0841 2 |
: 849      0842 2 | replymain_init ();
: 850      0843 2 |
: 851      0844 2 | If one of the broadcast qualifiers is used, call the broadcast routine
: 852      0845 2 |
: 853      0846 2 | IF cli$present (ascid_ALL)
: 854      0847 2 | OR
: 855      0848 2 |     cli$present (ascid_TERMINAL)
: 856      0849 2 | OR
: 857      0850 2 |     cli$present (ascid_USERNAME)
: 858      0851 2 | THEN
: 859      0852 2 |     RETURN replymain_broadcast ();
: 860      0853 2 |
: 861      0854 2 |
: 862      0855 2 | If enable or disable operator's terminal, call that routine
: 863      0856 2 |
: 864      0857 2 | IF cli$present (ascid_DISABLE)
: 865      0858 2 | OR

```

```

: 866 0859 2 cli$present (ascid_ENABLE)
: 867 0860 2 THEN
: 868 0861 2 RETURN replymain_oprenable ();
: 869 0862 2
: 870 0863 2
: 871 0864 2 : If a logfile request, dispatch to the logfile action routine
: 872 0865 2
: 873 0866 2 status = cli$present (ascid_LOG);
: 874 0867 2 IF .status ! We have a /LOG
: 875 0868 2 OR
: 876 0869 2 .status EQL cli$_negated ! We have a /NOLOG
: 877 0870 2 THEN
: 878 0871 2 RETURN replymain_logfile ();
: 879 0872 2
: 880 0873 2
: 881 0874 2 : If a request for status, do it
: 882 0875 2
: 883 0876 2 IF cli$present (ascid_STATUS)
: 884 0877 2 THEN
: 885 0878 2 RETURN replymain_status ();
: 886 0879 2
: 887 0880 2
: 888 0881 2 : Otherwise, we assume it is one of the miscellaneous replies to requests,
: 889 0882 2 as in /ABORT, /BLANK_TAPE, /INITIALIZE_TAPE, /PENDING or /TO.
: 890 0883 2
: 891 0884 2 RETURN replymain_reply ();
: 892 0885 2
: 893 0886 1 END; ! End of replymain_main

```

```

                                .EXTRN CLIS_NEGATED
                                .ENTRY REPLYMAIN MAIN, Save R2
FE6A 52 00000000G 00 0004 00000 MOVAB CLISPRESNT, R2 : 0802
CF 00 9E 00002 CALLS #0, REPLYMAIN_INIT : 0842
0000' CF 9F 0000E PUSHAB ASCID_ALL : 0846
62 01 FB 00012 CALLS #1, C[ISPRESNT
14 50 E8 00015 BLBS R0, 1$
0000' CF 9F 00018 PUSHAB ASCID_TERMINAL : 0848
62 01 FB 0001C CALLS #1, C[ISPRESNT
0A 50 E8 0001F BLBS R0, 1$
0000' CF 9F 00022 PUSHAB ASCID_USERNAME : 0850
62 01 FB 00026 CALLS #1, C[ISPRESNT
06 50 E9 00029 BLBC R0, 2$
FA43 CF 00 FB 0002C 1$: CALLS #0, REPLYMAIN_BROADCAST : 0852
0000' CF 04 00031 RET
62 01 FB 00032 2$: PUSHAB ASCID_DISABLE : 0857
0A 50 E8 00039 BLBS R0, 3$
0000' CF 9F 0003C PUSHAB ASCID_ENABLE : 0859
62 01 FB 00040 CALLS #1, C[ISPRESNT
06 50 E9 00043 BLBC R0, 4$
0000V CF 00 FB 00046 3$: CALLS #0, REPLYMAIN_OPRENABLE : 0861
0000' CF 04 00048 RET
62 01 FB 00050 4$: PUSHAB ASCID_LOG : 0866
CALLS #1, C[ISPRESNT

```


OPCSREPLYMAIN
V04-000

REPLY command main module
replymain_main routine

M 10
16-Sep-1984 01:44:54
14-Sep-1984 12:30:54

VAX-11 Bliss-32 V4.0-742
[OPCOM.SRC]REPLYMAIN.B32;1

Page 31
(8)

OP
VO

00000000G	09	50	E8	00053	BLBS	STATUS, 5\$:	0867
	8F	50	D1	00056	CMPL	STATUS, #CLIS_NEGATED	:	0869
		06	12	0005D	BNEG	6\$:	
FF35	CF	00	FB	0005F 5\$:	CALLS	#0, REPLYMAIN_LOGFILE	:	0871
			04	00064	RET		:	
		0000'	CF	9F 00065 6\$:	PUSHAB	ASCID STATUS	:	0876
	62		01	FB 00069	CALLS	#1, CEISPRESENT	:	
	06		50	E9 0006C	BLBC	RO, 7\$:	
0000V	CF		00	FB 0006F	CALLS	#0, REPLYMAIN_STATUS	:	0878
				04 00074	RET		:	
0000V	CF		00	FB 00075 7\$:	CALLS	#0, REPLYMAIN_REPLY	:	0884
			04	0C07A	RET		:	0886

: Routine Size: 123 bytes, Routine Base: \$CODE\$ + 058C

```

895 0887 1 GLOBAL ROUTINE replymain_oprenable = %SBTTL 'replymain_oprenable'
896 0888 1 ++
897 0889 1 Functional description:
898 0890 1
899 0891 1 This routine controls enabling or disabling operator terminals.
900 0892 1
901 0893 1 Input:
902 0894 1
903 0895 1 None.
904 0896 1
905 0897 1 Implicit Input:
906 0898 1
907 0899 1 CLI parameters
908 0900 1
909 0901 1 Output:
910 0902 1
911 0903 1 None.
912 0904 1
913 0905 1 Implicit output:
914 0906 1
915 0907 1 None.
916 0908 1
917 0909 1 Side effects:
918 0910 1
919 0911 1 None.
920 0912 1
921 0913 1 Routine value:
922 0914 1
923 0915 1 None.
924 0916 1 --
925 0917 1
926 0918 2 BEGIN ! Start of replymain_oprenable
927 0919 2
928 0920 2 REGISTER
929 0921 2 mlen, ! Output message length
930 0922 2 mptr : $ref_bvector; ! Output message pointer
931 0923 2
932 0924 2 LOCAL
933 0925 2 text : $dyn_str_desc, ! Dynamic string descr for message text
934 0926 2 message : $bblock [128], ! Buffer to build message
935 0927 2 message_desc : VECTOR [2, LONG] PRESET ([1] = message),
936 0928 2 idx,
937 0929 2 status,
938 0930 2 type_keyword;
939 0931 2
940 0932 2 Initialize the message
941 0933 2
942 0934 2 NOTE: We are using an internal interface to OPCOM which is subject to change!
943 0935 2
944 0936 2 CH$FILL (0, opc$k_oprenable_min_size, message); ! Init all fixed fields to zero
945 0937 2 message [opc$b_rqstcode] = opc$x_oprenable;
946 0938 2 message [opc$b_scope] = opc$k_system;
947 0939 2 IF cli$present(ascid_DISABLE)
948 0940 2 THEN
949 0941 3 BEGIN
950 0942 3 $bblock [message [opc$l_rq_options], opc$v_disable] = true;
951 0943 3 type_keyword = ascid_DISABLE;

```

```

952 0944 3   END
953 0945 2   ELSE
954 0946 3   BEGIN
955 0947 3   type_keyword = ascid_ENABLE;
956 0948 3   IF NOT cli$present (ascid_TEMPORARY)
957 0949 3   THEN
958 0950 3   $bblock [message [opc$l_rq_options], opc$v_permoper] = true;
959 0951 2   END;
960 0952 2   :
961 0953 2   : Move the sending terminal name
962 0954 2   :
963 0955 2   mptr = message [opc$t_oprenable_opr];           ! Set output pointer to start of text area
964 0956 2   mlen = .dvi_terminal_len;                       ! Get length of terminal name
965 0957 2   mptr [0] = .mlen;                                 ! Store the ASCII length
966 0958 2   CH$MOVE (.mlen, dvi_terminal_buf, mptr [1]);   ! Append the name to the buffer
967 0959 2   message_desc [0] = $byteoffset (opc$t_oprenable_opr) + 1 + .mlen; ! Save total length
968 0960 2   :
969 0961 2   : Set the attention mask according to the appropriate qualifier
970 0962 2   :
971 0963 2   IF NOT cli$get_value (.type_keyword, text)
972 0964 2   THEN
973 0965 2   :
974 0966 2   : The qualifier is /ENABLE or /DISABLE without any keywords. Operate on all operators.
975 0967 2   :
976 0968 3   BEGIN
977 0969 3   message [opc$l_attnmask1] = known_attn_mask1;
978 0970 3   message [opc$l_attnmask2] = known_attn_mask2;
979 0971 3   END
980 0972 2   ELSE
981 0973 2   :
982 0974 2   : The qualifier is /xABLE=(...), set the bit for each specified operator
983 0975 2   :
984 0976 2   DO $bblock [message [opc$l_attnmask1], 0, share_lookup_oper_bit (text), 1, 0] = 1
985 0977 2   UNTIL NOT cli$get_value (.type_keyword, text);
986 0978 2   :
987 0979 2   : Send the message to OPCOM
988 0980 2   :
989 0981 3   IF NOT (status = $sndopr (msgbuf=message_desc))
990 0982 2   THEN
991 0983 2   $signal_stop (.status);
992 0984 2   :
993 0985 2   RETURN ss$_normal;
994 0986 1   END;

```

! End of replymain_oprenable

		03FC 00000		.ENTRY	REPLYMAIN_OPRENABLE, Save R2,R3,R4,R5,R6.-	: 0887
					R7,R8,R9	:
	59	00000000G	00 9E 00002	MOVAB	CLISGET VALUE, R9	:
	58	00000000G	00 9E 00009	MOVAB	CLISPRESENT, R8	:
	5E	FF74	CE 9E 00010	MOVAB	-140(SP), SP	:
F8	AD	020E0000	8F D0 00015	MOVL	#34471936, TEXT	: 0925
		FC	AD D4 0001D	CLRL	TEXT+4	:
			7E D4 00020	CLRL	MESSAGE_DESC	: 0927
04	AE	08	AE 9E 00022	MOVAB	MESSAGE, MESSAGE_DESC+4	:

1E	00	6E	00	2C	00027	MOVCS	#0, (SP), #0, #30, MESSAGE	: 0936
		08	AE		0002C			
		08	AE	010A	8F B0	MOVW	#266, MESSAGE	: 0937
				0000'	CF 9F	PUSHAB	ASCID_DISABLE	: 0939
		68	01	FB	00038	CALLS	#1, C[ISPRESENT	
		0B	50	E9	0003B	BLBC	RO, 1\$	
		0E	01	88	0003E	BISB2	#1, MESSAGE+6	: 0942
		57	CF	9E	00042	MOVAB	ASCID_DISABLE, TYPE_KEYWORD	: 0943
					13 11	BRB	2\$: 0939
		57	CF	9E	00049	MOVAB	ASCID_ENABLE, TYPE_KEYWORD	: 0947
					01 9F	PUSHAB	ASCID_TEMPORARY	: 0948
		68	01	FB	00052	CALLS	#1, C[ISPRESENT	
		04	50	E8	00055	BLBS	RO, 2\$	
		0E	02	88	00058	BISB2	#2, MESSAGE+6	: 0950
		50	AE	9E	0005C	MOVAB	MESSAGE+26, MPTR	: 0955
		56	CF	D0	00060	MOVL	DVI_TERMINAL_LEN, MLEN	: 0956
		60	56	90	00065	MOVW	MLEN, (MPTR)	: 0957
01	A0	0000'	CF	56	28	MOVCS	MLEN, DVI_TERMINAL_BUF, 1(MPTR)	: 0958
			6E	A6	9E	MOVAB	27(R6), MESSAGE_DESC	: 0959
				F8	AD	PUSHAB	TEXT	: 0963
					57 DD	PUSHL	TYPE_KEYWORD	
		69	02	FB	00078	CALLS	#2, C[ISGET_VALUE	
		0D	50	E8	0007B	BLBS	RO, 3\$	
		12	AE	00FF1FF	8F D0	MOVL	#16773631, MESSAGE+10	: 0969
					16 AE	CLRL	MESSAGE+14	: 0970
					18 11	BRB	5\$: 0963
					F8 AD	PUSHAB	TEXT	: 0976
					01 FB	CALLS	#1, SHARE_LOOKUP_OPER_BIT	
		0000G	CF	50	E2	BBSS	RO, MESSAGE+10, 4\$	
		12	AE	F8	AD	PUSHAB	TEXT	: 0977
					57 DD	PUSHL	TYPE_KEYWORD	
		69	02	FB	0009D	CALLS	#2, C[ISGET_VALUE	
		E8	50	E8	000A0	BLBS	RO, 3\$	
					7E D4	CLRL	-(SP)	: 0981
			04	AE	9F	PUSHAB	MESSAGE_DESC	
		00000000G	00	02	FB	CALLS	#2, SYS\$SNDOPR	
			0A	50	E8	BLBS	STATUS, 6\$	
					50 DD	PUSHL	STATUS	: 0983
		00000000G	00	01	FB	CALLS	#1, LIB\$STOP	
					04	RET		
			50	01	D0	MOVL	#1, RO	: 0985
					04	RET		: 0986

; Routine Size: 192 bytes, Routine Base: \$CODE\$ + 0607

```

: 996 0987 1 GLOBAL ROUTINE replymain_reply = %SBTTL 'replymain_reply'
: 997 0988 1 ++
: 998 0989 1 Functional description:
: 999 0990 1
: 1000 0991 1 This routine controls enabling or disabling operator terminals.
: 1001 0992 1
: 1002 0993 1 Input:
: 1003 0994 1
: 1004 0995 1 None.
: 1005 0996 1
: 1006 0997 1 Implicit Input:
: 1007 0998 1
: 1008 0999 1 CLI parameters
: 1009 1000 1
: 1010 1001 1 Output:
: 1011 1002 1
: 1012 1003 1 None.
: 1013 1004 1
: 1014 1005 1 Implicit output:
: 1015 1006 1
: 1016 1007 1 None.
: 1017 1008 1
: 1018 1009 1 Side effects:
: 1019 1010 1
: 1020 1011 1 None.
: 1021 1012 1
: 1022 1013 1 Routine value:
: 1023 1014 1
: 1024 1015 1 None.
: 1025 1016 1 --
: 1026 1017 1
: 1027 1018 2 BEGIN ! Start of replymain_reply
: 1028 1019 2
: 1029 1020 2 REGISTER
: 1030 1021 2 mlen, ! Output message length
: 1031 1022 2 mptr : $ref_bvector; ! Output message pointer
: 1032 1023 2
: 1033 1024 2 LOCAL
: 1034 1025 2 text : $dyn_str_desc, ! Dynamic string descr for message text
: 1035 1026 2 message : $bblock [2048], ! Buffer to build message
: 1036 1027 2 message_desc : VECTOR [2, LONG] PRESET ([1] = message),
: 1037 1028 2 idx,
: 1038 1029 2 status,
: 1039 1030 2 type_keyword;
: 1040 1031 2
: 1041 1032 2 Initialize the message
: 1042 1033 2
: 1043 1034 2 NOTE: We are using an internal interface to OPCOM which is subject to change!
: 1044 1035 2
: 1045 1036 2 CH$FILL (0, opc$k_reply_min_size, message); ! Init all fixed fields to zero
: 1046 1037 2 message [opc$b_rqstcode] = opc$x_reply;
: 1047 1038 2 message [opc$b_scope] = opc$k_system;
: 1048 1039 2
: 1049 1040 2 Find out which flavor of reply. The main routine calls us if it hasn't found something else, therefore
: 1050 1041 2 if it isn't one of ours we need to return the bad status.
: 1051 1042 2
: 1052 1043 2 SELECTONE cli$_present OF

```

```

: 1053      1044 2 SET
: 1054      1045 2 [cli$present (ascid_ABORT)] : BEGIN
: 1055      1046 2 message [opc$l_rq_options] = opc$_rqstabort;
: 1056      1047 2 type_keyword = ascid_ABORT;
: 1057      1048 2 END;
: 1058      1049 2 [cli$present (ascid_BLANK_TAPE)] : BEGIN
: 1059      1050 2 message [opc$l_rq_options] = opc$_blanktape;
: 1060      1051 2 type_keyword = ascid_BLANK_TAPE;
: 1061      1052 2 END;
: 1062      1053 2 [cli$present (ascid_INITIALIZE_TAPE)] : BEGIN
: 1063      1054 2 message [opc$l_rq_options] = opc$_initape;
: 1064      1055 2 type_keyword = ascid_INITIALIZE_TAPE;
: 1065      1056 2 END;
: 1066      1057 2 [cli$present (ascid_PENDING)] : BEGIN
: 1067      1058 2 message [opc$l_rq_options] = opc$_rqstpend;
: 1068      1059 2 type_keyword = ascid_PENDING;
: 1069      1060 2 END;
: 1070      1061 2 [cli$present (ascid_T0)] : BEGIN
: 1071      1062 2 message [opc$l_rq_options] = opc$_rqstcmplte;
: 1072      1063 2 type_keyword = ascid_T0;
: 1073      1064 2 END;
: 1074      1065 2 [OTHERWISE] : RETURN cli$_ivverb;
: 1075      1066 2 TES;
: 1076      1067 2 :
: 1077      1068 2 : Move the request ID to the message
: 1078      1069 2 :
: 1079      1070 3 IF NOT (status = cli$get_value (.type_keyword, text))
: 1080      1071 2 THEN
: 1081      1072 2 $signal_stop (.status); ! This is a required entity!
: 1082      1073 3 IF NOT (status = ots$cv_t_l (text, message [opc$l_rqstid]))
: 1083      1074 2 THEN
: 1084      1075 2 $signal_stop (opc$_valuerr, 1, text, .status);
: 1085      1076 2 :
: 1086      1077 2 : Move the sending terminal name
: 1087      1078 2 :
: 1088      1079 2 mptr = message [opc$t_reply_opr]; ! Set output pointer to start of text area
: 1089      1080 2 mlen = .dvi_terminal_len; ! Get length of terminal name
: 1090      1081 2 mptr [0] = .mlen; ! Store the ASCII length
: 1091      1082 2 CHSMOVE (.mlen, dvi_terminal_buf, mptr [1]); ! Append the name to the buffer
: 1092      1083 2 message_desc [0] = $byteoffset (opc$t_reply_opr) + 1 + .mlen; ! Save total length
: 1093      1084 2 mptr = .mptr + 1 + .mlen;
: 1094      1085 2 :
: 1095      1086 2 :
: 1096      1087 2 : Move the reply text, if any
: 1097      1088 2 :
: 1098      1089 2 cli$get_value (ascid_P1, text); ! Get the parameter
: 1099      1090 2 (mptr [0]) <0,16,0> = .text [dsc$w_length]; ! Store 16-bit length of text
: 1100      1091 2 message_desc [0] = .message_desc [0] + 2 + .text [dsc$w_length]; ! Add text plus length word to total length
: 1101      1092 2 IF .text [dsc$w_length] GTR 0 ! If a message came in, move it to the buffer after the len
: 1102      1093 2 THEN
: 1103      1094 2 CHSMOVE (.text [dsc$w_length], .text [dsc$a_pointer], mptr [2]);
: 1104      1095 2 :
: 1105      1096 2 : Send the message to OPCOM
: 1106      1097 2 :
: 1107      1098 3 IF NOT (status = $sndopr (msgbuf=message_desc))
: 1108      1099 2 THEN
: 1109      1100 2 $signal_stop (.status);

```

: 1110
: 1111
: 1112
1101 2
1102 2 RETURN ss\$_normal;
1103 1 END;

! End of replymain_reply

20 00

```

OFFC 00000
SB 00000000G 00 9E 00002
SA 00000000G 00 9E 00009
S9 0000' CF 9E 00010
SE F7F4 CE 9E 00015
F8 AD 020E0000 8F D0 0001A
FC AD D4 00022
7E D4 00025
04 AE 08 AE 9E 00027
6E 00 2C 0002C
08 AE 08 AE 00031
010D 8F B0 00033
52 00000000G 8F D0 00039
59 DD 00040
6A 01 FB 00042
50 52 D1 00045
0D 12 00048
OE AE 0005801C 8F D0 0004A
53 69 9E 00052
2C A9 9F 00057 1$:
6A 01 FB 0005A
50 52 D1 0005D
0E 12 00060
OE AE 000581E3 8F D0 00062
53 2C A9 9E 0006A
57 11 0006E
6A 64 A9 9F 00070 2$:
50 0' FB 00073
52 D1 00076
0E 12 00079
OE AE 000581D3 8F D0 0007B
53 64 A9 9E 00083
00B8 3E 11 00087
3$:
6A 01 FB 0008D
50 52 D1 00090
0F 12 00093
OE AE 00058021 8F D0 00095
53 00B8 C9 9E 0009D
23 11 000A2
012C C9 9F 000A4 4$:
6A 01 FB 000AB
50 52 D1 000AB
0F 12 000AE
OE AE 00058029 8F D0 000B0
53 012C C9 9E 000B8

```

```

.EXTRN CLIS_PRESENT, CLIS_IVVERB
.EXTRN OTSS$CVT_TI_L
.ENTRY REPLYMAIN_REPLY, Save R2,R3,R4,R5,R6,R7,R8,-; 0987
R9,R10,R11
MOVAB CLIS$GET_VALUE, R11
MOVAB CLIS$PRESENT, R10
MOVAB ASCID_ABORT, R9
MOVAB -20607(SP), SP
MOVL #34471936, TEXT 1025
CLRL TEXT+4
CLRL MESSAGE_DESC 1027
MOVAB MESSAGE, MESSAGE_DESC+4
MOVCS #0, (SP), #0, #32, MESSAGE 1036
MOVW #269, MESSAGE 1037
MOVL #CLIS_$PRESENT, R2 1043
PUSHL R9 1045
CALLS #1, CLIS$PRESENT
CMPL R2, R0
BNEQ 1$
MOVL #360476, MESSAGE+6 1046
MOVAB ASCID_ABORT, TYPE_KEYWORD 1047
BRB 6$ 1043
PUSHAB ASCID_BLANK_TAPE 1049
CALLS #1, CLIS$PRESENT
CMPL R2, R0
BNEQ 2$
MOVL #360931, MESSAGE+6 1050
MOVAB ASCID_BLANK_TAPE, TYPE_KEYWORD 1051
BRB 6$ 1043
PUSHAB ASCID_INITIALIZE_TAPE 1053
CALLS #1, CLIS$PRESENT
CMPL R2, R0
BNEQ 3$
MOVL #360915, MESSAGE+6 1054
MOVAB ASCID_INITIALIZE_TAPE, TYPE_KEYWORD 1055
BRB 6$ 1043
PUSHAB ASCID_PENDING 1057
CALLS #1, CLIS$PRESENT
CMPL R2, R0
BNEQ 4$
MOVL #360481, MESSAGE+6 1058
MOVAB ASCID_PENDING, TYPE_KEYWORD 1059
BRB 6$ 1043
PUSHAB ASCID_TO 1061
CALLS #1, CLIS$PRESENT
CMPL R2, R0
BNEQ 5$
MOVL #360489, MESSAGE+6 1062
MOVAB ASCID_TO, TYPE_KEYWORD 1063

```

			08	11	000BD				BRB	6\$		1043
		50	00000000G	8F	DO	000BF	5\$:		MOVL	#CLIS_IVVERB, R0		1063
					04	000C6			RET			
				F8	AD	9F	6\$:		PUSHAB	TEXT		1070
					53	DD			PUSHL	TYPE_KEYWORD		
		6B			02	FB			CALLS	#2, CLISGET_VALUE		
		58			50	DO			MOVL	R0, STATUS		
		7B			58	E9			BLBC	STATUS, 9\$		
					1A	AE			PUSHAB	MESSAGE+18		1073
				F8	AD	9F			PUSHAB	TEXT		
		00000000G	00		02	FB			CALLS	#2, OTSSCVT_TI_L		
			58		50	DO			MOVL	R0, STATUS		
			15		58	EB			BLBS	STATUS, 7\$		1075
					58	DD			PUSHL	STATUS		
				F8	AD	9F			PUSHAB	TEXT		
					01	DD			PUSHL	#1		
		00000000G	00	0005825C	8F	DD			PUSHL	#361052		
					04	FB			CALLS	#4, LIB\$STOP		
					04	000FC			RET			
		57			22	AE	7\$:		MOVAB	MESSAGE+26, MPTR		1079
		56			0000'	CF			MOVL	DVI_TERMINAL_LEN, MLEN		1080
		67				56			MOVAB	MLEN, (MPTR)		1081
	01	A7	0000'			56			MOVAB	MLEN, DVI_TERMINAL_BUF, 1(MPTR)		1082
						28			MOVAB	27(R6), MESSAGE_DESC		1083
					1B	A6			MOVAB	1(MLEN)(MPTR), MPTR		1084
					01	A647			MOVAB	TEXT		1089
					F8	AD			PUSHAB	ASCID P1		
					00A8	C9			PUSHAB	TEXT		
						02			CALLS	#2, CLISGET_VALUE		
		6B			F8	AD			MOVW	TEXT, (MPTR)		1090
		67			F8	AD			MOVZWL	TEXT, R0		1091
		50				3C			ADDL2	MESSAGE_DESC, R0		
		50			6E	C0			MOVAB	2(R0), MESSAGE_DESC		
		6E				9E			TSTW	TEXT		1092
					F8	AD			BEQL	8\$		
						07			MOVAB	TEXT, @TEXT+4, 2(MPTR)		1094
	02	A7	FC	BD	F8	AD			CLRL	-(SP)		1098
						13			PUSHAB	MESSAGE_DESC		
						D4			CALLS	#2, SYS\$NDOPR		
					04	AE			MOVL	R0, STATUS		
		00000000G	00			02			BLBS	STATUS, 10\$		
			58			50			PUSHL	STATUS		1100
			0A			58			CALLS	#1, LIB\$STOP		
		00000000G	00			01			RET			
						04			RET			
						01			MOVL	#1, R0		1102
						04			RET			1103
						04						

: Routine Size: 350 bytes, Routine Base: \$CODE\$ + 06C7


```

: 1114      1104 1 GLOBAL ROUTINE replymain_status =                %SBTTL 'replymain_status'
: 1115      1105 1
: 1116      1106 1  Function-1 description:
: 1117      1107 1
: 1118      1108 1      This routine requests a display of status
: 1119      1109 1
: 1120      1110 1  Input:
: 1121      1111 1
: 1122      1112 1      None.
: 1123      1113 1
: 1124      1114 1  Implicit Input:
: 1125      1115 1
: 1126      1116 1      CLI parameters
: 1127      1117 1
: 1128      1118 1  Output:
: 1129      1119 1
: 1130      1120 1      None.
: 1131      1121 1
: 1132      1122 1  Implicit output:
: 1133      1123 1
: 1134      1124 1      None.
: 1135      1125 1
: 1136      1126 1  Side effects:
: 1137      1127 1
: 1138      1128 1      None.
: 1139      1129 1
: 1140      1130 1  Routine value:
: 1141      1131 1
: 1142      1132 1      None.
: 1143      1133 1  --
: 1144      1134 1
: 1145      1135 2 BEGIN                                           ! Start of replymain_status
: 1146      1136 2
: 1147      1137 2 REGISTER
: 1148      1138 2     mlen,                                           ! Output message length
: 1149      1139 2     mptr      : $ref_bvector;                       ! Output message pointer
: 1150      1140 2
: 1151      1141 2 LOCAL
: 1152      1142 2     message      : $bblock [128],                    ! Buffer to build message
: 1153      1143 2     message_desc : VECTOR [2, LONG] PRESET ([1] = message),
: 1154      1144 2     status;
: 1155      1145 2
: 1156      1146 2     Initialize the message
: 1157      1147 2
: 1158      1148 2     NOTE: We are using an internal interface to OPCOM which is subject to change!
: 1159      1149 2
: 1160      1150 2     CH$FILL (0, opc$k_status_min_size, message);      ! Init all fixed fields to zero
: 1161      1151 2     message [opc$b_rqstcode] = opc$x_status;
: 1162      1152 2     message [opc$b_scope] = opc$k_system;
: 1163      1153 2
: 1164      1154 2     Move the sending terminal name
: 1165      1155 2
: 1166      1156 2     mptr = message [opc$t_status_opr];                       ! Set output pointer to start of text area
: 1167      1157 2     mlen = dvi_terminal_len;                                   ! Get length of terminal name
: 1168      1158 2     mptr [0] = .mlen;                                       ! Store the ASCII length
: 1169      1159 2     CH$MOVE (.mlen, dvi_terminal_buf, mptr [1]);       ! Append the name to the buffer
: 1170      1160 2     message_desc [0] = $byteoffset (opc$t_status_opr) + 1 + .mlen; ! Save total length

```

```

: 1171      1161      2      |
: 1172      1162      2      | Send the message to OPCOM
: 1173      1163      2      |
: 1174      1164      2      | IF NOT (status = $sndopr (msgbuf=message_desc))
: 1175      1165      2      | THEN
: 1176      1166      2      |     $signal_stop (.status);
: 1177      1167      2      |
: 1178      1168      2      | RETURN ss$_normal;
: 1179      1169      1      | END;

```

! End of replymain_status

					007C 0000	.ENTRY	REPLYMAIN_STATUS, Save R2,R3,R4,R5,R6	: 1104
		5E	FF7C	CE	9E 00002	MOVAB	-132(SP), -SP	: 1143
				7E	D4 00007	CLRL	MESSAGE_DESC	: 1150
1E		04		AE	9E 00009	MOVAB	MESSAGE, MESSAGE_DESC+4	: 1151
	00			00	2C 0000E	MOVCS	#0, (SP), #0, #30, MESSAGE	: 1156
				08	AE 00013			: 1157
		08		AE	010F 8F B0 00015	MOVW	#271, MESSAGE	: 1158
				50	22 AE 9E 0001B	MOVAB	MESSAGE+26, MPTR	: 1159
				56	0000' CF D0 0001F	MOVL	DVI TERMINAL_LEN, MLEN	: 1160
				60	56 90 00024	MOVB	MLEN, (MPTR)	: 1164
	01	A0	0000'	CF	56 28 00027	MOVCS	MLEN, DVI TERMINAL_BUF, 1(MPTR)	: 1166
				6E	1B A6 9E 0002E	MOVAB	27(R6), MESSAGE_DESC	: 1168
					7E D4 00032	CLRL	-(SP)	: 1169
				04	AE 9F 00034	PUSHAB	MESSAGE_DESC	
		00000000G	00	02	FB 00037	CALLS	#2, SYS\$SNDOPR	
			0A	50	E8 0003E	BLBS	STATUS, 1\$	
		00000000G	00	50	DD 00041	PUSHL	STATUS	
				01	FB 00043	CALLS	#1, LIB\$STOP	
					04 0004A	RET		
			50	01	D0 0004B 1\$:	MOVL	#1, R0	
				04	0004E	RET		

: Routine Size: 79 bytes. Routine Base: \$CODE\$ + 0825

: 1181 1170 1 END
: 1182 1171 0 ELUDOM

! End of REPLYMAIN

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	276	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	366	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	2164	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	42	0	1000	00:01.8
_\$255\$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1	633	77	12	43	00:00.9

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:REPLYMAIN/OBJ=OBJ\$:REPLYMAIN MSRCS:REPLYMAIN/UPDATE=(ENHS:REPLYMAIN)

: Size: 2164 code + 642 data bytes
: Run Time: 00:41.2
: Elapsed Time: 02:14.6
: Lines/CPU Min: 1705
: Lexemes/CPU-Min: 22144
: Memory Used: 272 pages
: Compilation Complete

0291 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

