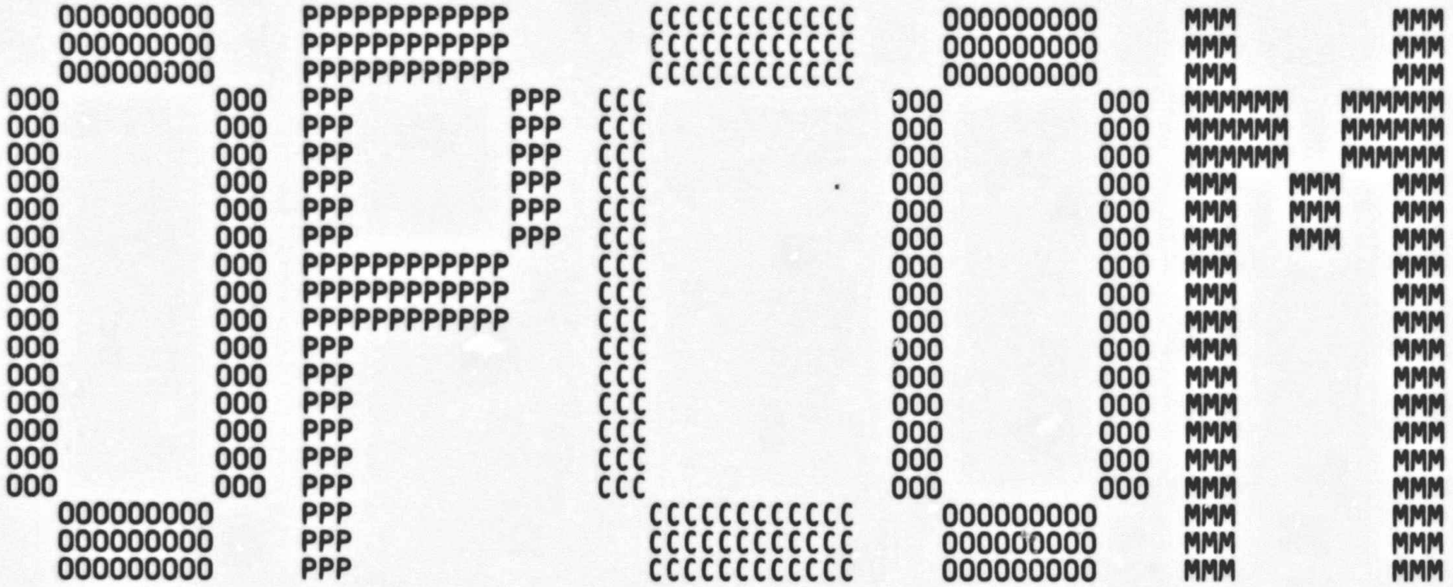

FILEID**OPCDEFTMP

```
  OOOOOO    PPPPPPPP      CCCCCCCC  DDDDDDDD   EEEEEEEEEE  FFFFFFFFFF  TTTTTTTTTT  MM      MM  PPPPPPPP
  OOOOOO    PPPPPPPP      CCCCCCCC  DDDDDDDD   EEEEEEEEEE  FFFFFFFFFF  TTTTTTTTTT  MM      MM  PPPPPPPP
OO      OO  PP      PP  CC        DD      DD  EE          FF              TT      MMMM  MMMM  PP      PP
OO      OO  PP      PP  CC        DD      DD  EE          FF              TT      MMMM  MMMM  PP      PP
OO      OO  PP      PP  CC        DD      DD  EE          FF              TT      MM MM MM MM  PP      PP
OO      OO  PP      PP  CC        DD      DD  EE          FF              TT      MM  MM  MM  PP      PP
OO      OO  PPPPPPPP    CC        DD      DD  EEEEEEEE    FFFFFFFF        TT      MM      MM  PPPPPPPP
OO      OO  PPPPPPPP    CC        DD      DD  EEEEEEEE    FFFFFFFF        TT      MM      MM  PPPPPPPP
OO      OO  PP          CC        DD      DD  EE          FF              TT      MM      MM  PP
OO      OO  PP          CC        DD      DD  EE          FF              TT      MM      MM  PP
OO      OO  PP          CC        DD      DD  EE          FF              TT      MM      MM  PP
OO      OO  PP          CC        DD      DD  EE          FF              TT      MM      MM  PP
  OOOOOO    PP            CCCCCCCC  DDDDDDDD   EEEEEEEEEE  FF              TT      MM      MM  PP
  OOOOOO    PP            CCCCCCCC  DDDDDDDD   EEEEEEEEEE  FF              TT      MM      MM  PP
```

```
LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
{ OPCDEFTMP.SDL - temporary system definition file for OPCOM internal structures
{
{ Version:      'V04-000'
{
{*****************************************************************************
{*                                                                           *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
{*  ALL RIGHTS RESERVED.                                                     *
{*                                                                           *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
{*  TRANSFERRED.                                                             *
{*                                                                           *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
{*  CORPORATION.                                                             *
{*                                                                           *
{*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
{*                                                                           *
{*****************************************************************************

{++
{
{ FACILITY:  OPCOM  -  Operator Communications
{
{ ABSTRACT:
{
{       This file contains the SDL source for OPCOM internal structure
{       definitions.  These are temporary structures which are likely
{       to change soon.
{
{ AUTHOR:  CW Hobbs      CREATION DATE:  28-Jun-1983
{
{ MODIFICATION HISTORY:
{
{       V03-004 CWH3169         CW Hobbs                        5-May-1984
{               Second pass for cluster-wide OPCOM:
{                - Add shutdown cluster message type and structure.
{                - Bump software version number
{
{       V03-003 RSH0111         R. Scott Hanna          12-Mar-1984
{               Define a symbol for the maximum formatted message
{               size. (OPC$K_MAXMESSAGE)
{
{       V03-002 CWH3002         CW Hobbs                16-Sep-1983
{               Add CLM__CLUMBX message type
{
{--
module OPCDEFTMP;
```

```
57  /*
58  /* Operator scope definitions, do one set with "K" tags and one set with
59  /* "C" tags.
60  /*
61
62  constant (
63      SYSTEM,
64      GROUP,
65      USER,
66      UNSPEC
67      ) equals 1 increment 1 prefix OPC$ tag "K";
68
69  constant (
70      SYSTEM,
71      GROUP,
72      USER,
73      UNSPEC
74      ) equals 1 increment 1 prefix OPC$ tag "C";
75
76  /*
77  /* The version number constant loosely describes the generation number of
78  /* OPCOM.  This number would be manually bumped at significant times in the
79  /* development cycle of OPCOM.  It should be used to detect (and hopefully
80  /* cope) with the situation of different versions of OPCOM executing on
81  /* different nodes of a cluster.  OPCOM cluster-wide data structures also
82  /* have version numbers.
83  /*
84  constant OPC$K_SW_VERSION equals 9;
85
86  /*
87  /* Miscellaneous numbers
88  /*
89  constant OPC$K_MAXREAD   equals 2560;
90  constant OPC$K_MAXMESSAGE equals 2048;
91  constant OPC$K_COMHDRSIZ equals 38;
92  constant OPC$_OPCOMERROR equals 99999;          /* New error message
93
94  /*
95  /* Define message codes for new format messages
96  /*
97  constant (
98      /*
99      /* New format analogs to old messages.  These might be referenced by other facilities,
100     /* so changing the values requires a system build.
101     /*
102     OPRENABLE,
103     LOGFILE,
104     REQUEST,
105     REPLY,
106     CANCEL,
107     STATUS,
108     SHUTDOWN,
109     TIMESTAMP,
110     SECURITY,
111     /*
112     /* Request codes for cluster communication messages
113     /*
```



```
114     CLUSMSG,
115     /*
116     /* Define special debugging code
```

```
117         /*
118         DEBUG,
119         /*
120         /* Connection manager messages
121         /*
122         CNXMAN,
123         /*
124         /* Dummy code to receive highest legal value + 1
125         /*
126         REQUEST_END_MARK
127         ) equals 10 increment 1 prefix OPC$_ tag "X";
128
129
130     /*
131     /* Define secondary message codes for inter-node cluster messages
132     /*
133     constant (
134         /*
135         ACKNOWLEDGEMENT,     /* Response to acknowledge request
136         ACKNOWLEDGE_PLEASE,  /* Request for remote node to announce itself
137         CANCEL,              /* Explicit cancel of request
138         CHECK_OPERATOR,      /* Make sure this operator is in the database
139         CHECK_REQUEST,       /* Make sure a request is in the database
140         CLUMBX,              /* Cluster mailbox message passed from cnxman
141         CLUSTER,             /* Cluster status change report
142         DEVICE,              /* Device message (on-line, off-line etc)
143         IMP_CANCEL,          /* Implicitly cancel a request
144         IMP_DISABLE,         /* Implicitly disable an operator
145         OPRENABLE,           /* Tell everyone else to enable or disable an operator
146         REPLY,               /* REPLY /PEND etc command
147         REPLY_COMPLETE,      /* Operator request completed by operator
148         REQUEST,             /* Operator request
149         RPYBRD,              /* Message from OPCOM to remotes, info for cluster REPLY /TERM, etc
150         RPYBRD_LOCAL,        /* Broadcast message from REPLY to OPCOM on local node
151         RPYNOT,              /* Reply notifications
152         SECURITY,            /* Security alarm from remote
153         SHUTDOWN,            /* Shut down operations
154         /*
155         /* Dummy code to receive highest legal value + 1
156         /*
157         REQUEST_END_MARK
158         ) equals 1 increment 1 prefix CLM_ tag '"';
159
```



```
160     /*
161     /* Temporary macro definitions for macros that will later be
162     /* defined in the $OPCDEF macro.  These are the offsets for the
163     /* various message formats.
164     /*
165
166     /*
167     /* Define the request header.  All messages (with the exception
168     /* of the device on/offline messages) have a common header.
169     /*
170
171     aggregate HEADER_MESSAGE structure prefix OPC$ fill;
172         RQSTCODE    byte unsigned;          /* Request code
173         SCOPE       byte unsigned;          /* Request SCOPE
174         OPTIONS     longword unsigned;      /* Request independent option bits.
175         RQ_OPTIONS  longword unsigned;      /* Request dependent options
176         ATTNMASK1   longword unsigned;      /* Attention mask part 1
```

```
177         ATTNMASK2   longword unsigned;      /* Attention mask part 2
178         RQSTID      longword unsigned;      /* User specified request id #
179         UIC         longword unsigned;      /* UIC of requestor
180
181         constant HDR_SIZE equals .;         /* Size of common header
182         end HEADER_MESSAGE;
183
184     /*
185     /* Option bits are carried around inside various structures.  Therefore, it
186     /* is more convenient to define them against the start of a longword, rather
187     /* than as a byte offset inside a structure.
188     /*
189     aggregate HEADER_OPTIONS structure longword unsigned prefix OPC$ fill;
190
191         /*
192         /* Define request independent option longword and bits.
193         /*
194         NOLOG       bitfield mask;          /* Do not log the action
195         NOBRD       bitfield mask;          /* Do not broadcast
196
197         end HEADER_OPTIONS;
198
```



```
199     /*
200     /* Define OPRENABLE message fields.
201     /*
202
203     aggregate OPRENABLE_MESSAGE structure prefix OPC$ fill;
204
205         OPRENABLE_FILL      byte dimension OPC$K_HDR_SIZE fill;
206
207         /*
208         /* Define place for the trailer message
209         /*
210         OPRENABLE_OPR       character length 0;     /* Start of oper dev name
211         constant OPRENABLE_MIN_SIZE equals . + 4;   /* Min message size header + 4
212
213         end OPRENABLE_MESSAGE;
214
215     aggregate OPRENABLE_OPTIONS structure longword unsigned prefix OPC$ fill;
216
217         /*
218         /* Define request dependent option bits.
219         /*
220         DISABLE     bitfield mask;
221         PERMOPER    bitfield mask;
222         NOREMIND    bitfield mask;
223
224         end OPRENABLE_OPTIONS;
225
226
227     /*
228     /* Define LOGFILE message fields.
229     /*
230
231     aggregate LOGFILE_MESSAGE structure prefix OPC$ fill;
232
233         LOGFILE_FILL        byte dimension OPC$K_HDR_SIZE fill;     /* Skip to request dependent options
234
235         /*
236         /* Define place for the trailer message
```

```
237         /*
238         LOGFILE_OPR character length 0;             /* Start of oper dev name
239         constant LOGFILE_MIN_SIZE equals . + 4;     /* Min message size header + 4
240
241         end LOGFILE_MESSAGE;
242
243     aggregate LOGFILE_OPTIONS structure longword unsigned prefix OPC$ fill;
244
245         /*
246         /* Define request dependent option bits.
247         /*
248         INITLOG     bitfield mask;
249         CLOSELOG    bitfield mask;
250         DISABLOG    bitfield mask;
251         ENABLOG     bitfield mask;
252
253         end LOGFILE_OPTIONS;
254
```



```
255     /*
256     /* Define REQUEST message fields.
257     /*
258
259     aggregate REQUEST_MESSAGE structure prefix OPC$ fill;
260
261         REQUEST_FILL        byte dimension OPC$K_HDR_SIZE fill;
262
263         /*
264         /* Define place for the trailer message length and text
265         /*
266         REQUEST_LENGTH      word unsigned;          /* Length of text
267         constant REQUEST_MIN_SIZE equals .;         /* Min message size
268         REQUEST_TEXT        character length 0;     /* Start of text
269
270         end REQUEST_MESSAGE;
271
272
273     /*
274     /* Define SECURITY message fields.
275     /*
276
277     aggregate SECURITY_MESSAGE structure prefix OPC$ fill;
278
279         SECURITY_FILL       byte dimension OPC$K_HDR_SIZE fill;
280
281         /*
282         /* Define place for the trailer message length and text
283         /*
284         SECURITY_LENGTH     word unsigned;          /* Length of text
285         constant SECURITY_MIN_SIZE equals .;        /* Min message size
286         SECURITY_TEXT       character length 0;     /* Start of text
287
288         end SECURITY_MESSAGE;
289
290
291     /*
292     /* Define REPLY message fields.
293     /*
294
295     aggregate REPLY_MESSAGE structure prefix OPC$ fill;
296
```

```
297         REPLY_FILL  byte dimension OPC$K_HDR_SIZE fill;
298
299         /*
300         /* After the ASCIC operator device name comes the counted (word
301         /* size count) of the reply text.  The text does not have to be
302         /* present.  The address of the count and the text itself can
303         /* be computed at run time.  The minimum size is the header, plus 4
304         /* for the device and 2 for the count.
305         /*
306         REPLY_OPR   character length 0;             /* Start of text
307         constant REPLY_MIN_SIZE equals . + 4 + 2;   /* Min message size
308
309         end REPLY_MESSAGE;
310
311
```



```
312     /*
313     /* Define STATUS message fields.
314     /*
315
316     aggregate STATUS_MESSAGE structure prefix OPC$ fill;
317
318         STATUS_FILL byte dimension OPC$K_HDR_SIZE fill;
319
320         /*
321         /* Define place for the operator device name.
322         /*
323         STATUS_OPR  character length 0;             /* Start of text
324         constant STATUS_MIN_SIZE equals . + 4;      /* Min message size header + 4
325
326         end STATUS_MESSAGE;
327
```



```
328     /*
329     /* Define TIMESTAMP message fields.
330     /*
331
332     aggregate TIMESTAMP_MESSAGE structure prefix OPC$ fill;
333
334         TIMESTAMP_FILL      byte dimension OPC$K_HDR_SIZE fill;
335
336         /*
337         /* Define the minimum length, no special fields
338         /*
339         constant TIMESTAMP_MIN_SIZE equals .;       /* Min message size
340
341         end TIMESTAMP_MESSAGE;
342
343
344     /*
345     /* Define SHUTDOWN message fields
346     /*
347
348     aggregate SHUTDOWN_MESSAGE structure prefix OPC$ fill;
349
350         SHUTDOWN_FILL       byte dimension OPC$K_HDR_SIZE fill;
351
352         /*
353         /* Define the minimum length, no special fields
```

```
354         /*
355         constant SHUTDOWN_MIN_SIZE equals .;        /* Min message size
356
357         end SHUTDOWN_MESSAGE;
358
359     aggregate SHUTDOWN_OPTIONS structure longword unsigned prefix OPC$ fill;
360
361         /*
362         /* Define request dependent option bits.
363         /*
364         CLUSTER     bitfield mask;
365
366         end SHUTDOWN_OPTIONS;
367
368
369     /*
370     /* Define CANCEL message fields
371     /*
372
373     aggregate CANCEL_MESSAGE structure prefix OPC$ fill;
374
375         CANCEL_FILL byte dimension OPC$K_HDR_SIZE fill;
376
377         /*
378         /* Define the minimum length, no special fields
379         /*
380         constant CANCEL_MIN_SIZE equals .;  /* Min message size
381
382         end CANCEL_MESSAGE;
383
384     aggregate CANCEL_OPTIONS structure longword unsigned prefix OPC$ fill;
```



```
385
386         /*
387         /* Define request dependent option bits.
388         /*
389         RQSTDONE    bitfield mask;
390
391         end CANCEL_OPTIONS;
392
393
394     end_module OPCDEFTMP;
```

LOGFILE
LIS

OPCOMDEF
LIS

OPCOMDATA
LIS

OPCOMINI
LIS

OPCOMUTIL
LIS

OPCOMLIB
LIS

OPCDEFTMP
LIS

OPCOMMAIN
LIS

OPCOMOLD
LIS

OPCOMRPLY
LIS

OPCCRASH
LIS

OPERUTIL
LIS

OPCOMROST
LIS