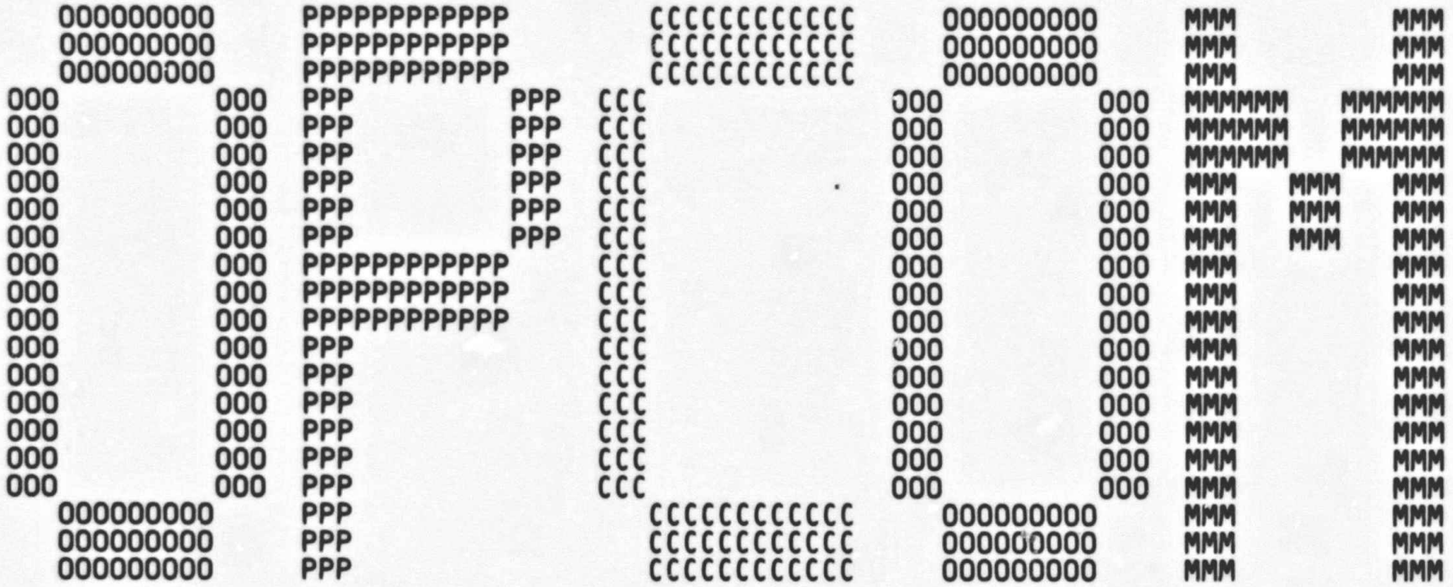

FILEID**OPCDEFTMP

```
  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  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
  OOOOOO    PP            CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FF              TT      MM      MM  PP
  OOOOOO    PP            CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FF              TT      MM      MM  PP

  SSSSSSSS  DDDDDDDD    LL
  SSSSSSSS  DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
  SSSSSS    DD      DD  LL
  SSSSSS    DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

....
....
....
....

OPC

/*
/*
/*
/*
agg

```
{ 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;
```

```
/*
/* Operator scope definitions, do one set with "K" tags and one set with
/* "C" tags.
/*

constant (
    SYSTEM,
    GROUP,
    USER,
    UNSPEC
    ) equals 1 increment 1 prefix OPC$ tag "K";

constant (
    SYSTEM,
    GROUP,
    USER,
    UNSPEC
    ) equals 1 increment 1 prefix OPC$ tag "C";

/*
/* The version number constant loosely describes the generation number of
/* OPCOM.  This number would be manually bumped at significant times in the
/* development cycle of OPCOM.  It should be used to detect (and hopefully
/* cope) with the situation of different versions of OPCOM executing on
/* different nodes of a cluster.  OPCOM cluster-wide data structures also
/* have version numbers.
/*
constant OPC$K_SW_VERSION equals 9;

/*
/* Miscellaneous numbers
/*
constant OPC$K_MAXREAD   equals 2560;
constant OPC$K_MAXMESSAGE equals 2048;
constant OPC$K_COMHDRSIZ equals 38;
constant OPC$_OPCOMERROR equals 99999;          /* New error message

/*
/* Define message codes for new format messages
/*
constant (
    /*
    /* New format analogs to old messages.  These might be referenced by other facilities,
    /* so changing the values requires a system build.
    /*
    OPRENABLE,
    LOGFILE,
    REQUEST,
    REPLY,
    CANCEL,
    STATUS,
    SHUTDOWN,
    TIMESTAMP,
    SECURITY,
    /*
    /* Request codes for cluster communication messages
```

```
    /*
    CLUSMSG,
    /*
    /* Define special debugging code
    /*
    DEBUG,
    /*
    /* Connection manager messages
    /*
    CNXMAN,
    /*
    /* Dummy code to receive highest legal value + 1
    /*
    REQUEST_END_MARK
    ) equals 10 increment 1 prefix OPC$_ tag "X";

/*
/* Define secondary message codes for inter-node cluster messages
/*
constant (
    /*
    ACKNOWLEDGEMENT,     /* Response to acknowledge request
    ACKNOWLEDGE_PLEASE,  /* Request for remote node to announce itself
    CANCEL,              /* Explicit cancel of request
    CHECK_OPERATOR,      /* Make sure this operator is in the database
    CHECK_REQUEST,       /* Make sure a request is in the database
    CLUMBX,              /* Cluster mailbox message passed from cnxman
    CLUSTER,             /* Cluster status change report
    DEVICE,              /* Device message (on-line, off-line etc)
    IMP_CANCEL,          /* Implicitly cancel a request
    IMP_DISABLE,         /* Implicitly disable an operator
    OPRENABLE,           /* Tell everyone else to enable or disable an operator
    REPLY,               /* REPLY /PEND etc command
    RPLY_COMPLETE,       /* Operator request completed by operator
    REQUEST,             /* Operator request
    RPYBRD,              /* Message from OPCOM to remotes, info for cluster REPLY /TERM, etc
    RPYBRD_LOCAL,        /* Broadcast message from REPLY to OPCOM on local node
    RPYNOT,              /* Reply notifications
    SECURITY,            /* Security alarm from remote
    SHUTDOWN,            /* Shut down operations
    /*
    /* Dummy code to receive highest legal value + 1
    /*
    REQUEST_END_MARK
    ) equals 1 increment 1 prefix CLM_ tag "";
```

```
/*
/* Temporary macro definitions for macros that will later be
/* defined in the $OPCDEF macro.  These are the offsets for the
/* various message formats.
/*

/*
/* Define the request header.  All messages (with the exception
/* of the device on/offline messages) have a common header.
/*

aggregate HEADER_MESSAGE structure prefix OPC$ fill;
    RQSTCODE    byte unsigned;          /* Request code
    SCOPE       byte unsigned;          /* Request SCOPE
    OPTIONS     longword unsigned;      /* Request independent option bits.
    RQ_OPTIONS  longword unsigned;      /* Request dependent options
    ATTNMASK1   longword unsigned;      /* Attention mask part 1
    ATTNMASK2   longword unsigned;      /* Attention mask part 2
    RQSTID      longword unsigned;      /* User specified request id #
    UIC         longword unsigned;      /* UIC of requestor

    constant HDR_SIZE equals .;         /* Size of common header
    end HEADER_MESSAGE;

/*
/* Option bits are carried around inside various structures.  Therefore, it
/* is more convenient to define them against the start of a longword, rather
/* than as a byte offset inside a structure.
/*
aggregate HEADER_OPTIONS structure longword unsigned prefix OPC$ fill;

    /*
    /* Define request independent option longword and bits.
    /*
    NOLOG       bitfield mask;          /* Do not log the action
    NOBRD       bitfield mask;          /* Do not broadcast

    end HEADER_OPTIONS;
```

```
/*
/* Define OPRENABLE message fields.
/*

aggregate OPRENABLE_MESSAGE structure prefix OPC$ fill;

    OPRENABLE_FILL      byte dimension OPC$K_HDR_SIZE fill;

    /*
    /* Define place for the trailer message
    /*
    OPRENABLE_OPR       character length 0;      /* Start of oper dev name
    constant OPRENABLE_MIN_SIZE equals . + 4;    /* Min message size header + 4

    end OPRENABLE_MESSAGE;

aggregate OPRENABLE_OPTIONS structure longword unsigned prefix OPC$ fill;

    /*
    /* Define request dependent option bits.
    /*
    DISABLE     bitfield mask;
    PERMOPER    bitfield mask;
    NOREMIND    bitfield mask;

    end OPRENABLE_OPTIONS;


/*
/* Define LOGFILE message fields.
/*

aggregate LOGFILE_MESSAGE structure prefix OPC$ fill;

    LOGFILE_FILL        byte dimension OPC$K_HDR_SIZE fill;     /* Skip to request dependent options

    /*
    /* Define place for the trailer message
    /*
    LOGFILE_OPR character length 0;             /* Start of oper dev name
    constant LOGFILE_MIN_SIZE equals . + 4;     /* Min message size header + 4

    end LOGFILE_MESSAGE;

aggregate LOGFILE_OPTIONS structure longword unsigned prefix OPC$ fill;

    /*
    /* Define request dependent option bits.
    /*
    INITLOG     bitfield mask;
    CLOSELOG    bitfield mask;
    DISABLOG    bitfield mask;
    ENABLOG     bitfield mask;

    end LOGFILE_OPTIONS;
```

```
/*
/* Define REQUEST message fields.
/*

aggregate REQUEST_MESSAGE structure prefix OPC$ fill;

    REQUEST_FILL        byte dimension OPC$K_HDR_SIZE fill;

    /*
    /* Define place for the trailer message length and text
    /*
    REQUEST_LENGTH      word unsigned;          /* Length of text
    constant REQUEST_MIN_SIZE equals .;         /* Min message size
    REQUEST_TEXT        character length 0;     /* Start of text

    end REQUEST_MESSAGE;


/*
/* Define SECURITY message fields.
/*

aggregate SECURITY_MESSAGE structure prefix OPC$ fill;

    SECURITY_FILL       byte dimension OPC$K_HDR_SIZE fill;

    /*
    /* Define place for the trailer message length and text
    /*
    SECURITY_LENGTH     word unsigned;          /* Length of text
    constant SECURITY_MIN_SIZE equals .;        /* Min message size
    SECURITY_TEXT       character length 0;     /* Start of text

    end SECURITY_MESSAGE;


/*
/* Define REPLY message fields.
/*

aggregate REPLY_MESSAGE structure prefix OPC$ fill;

    REPLY_FILL  byte dimension OPC$K_HDR_SIZE fill;

    /*
    /* After the ASCIC operator device name comes the counted (word
    /* size count) of the reply text.  The text does not have to be
    /* present.  The address of the count and the text itself can
    /* be computed at run time.  The minimum size is the header, plus 4
    /* for the device and 2 for the count.
    /*
    REPLY_OPR   character length 0;             /* Start of text
    constant REPLY_MIN_SIZE equals . + 4 + 2;   /* Min message size

    end REPLY_MESSAGE;
```

```
/*
/* Define STATUS message fields.
/*

aggregate STATUS_MESSAGE structure prefix OPC$ fill;

    STATUS_FILL byte dimension OPC$K_HDR_SIZE fill;

    /*
    /* Define place for the operator device name.
    /*
    STATUS_OPR  character length 0;             /* Start of text
    constant STATUS_MIN_SIZE equals . + 4;      /* Min message size header + 4

    end STATUS_MESSAGE;
```

```
/*
/* Define TIMESTAMP message fields.
/*
aggregate TIMESTAMP_MESSAGE structure prefix OPC$ fill;
    TIMESTAMP_FILL      byte dimension OPC$K_HDR_SIZE fill;
    /*
    /* Define the minimum length, no special fields
    /*
    constant TIMESTAMP_MIN_SIZE equals .;       /* Min message size
    end TIMESTAMP_MESSAGE;

/*
/* Define SHUTDOWN message fields
/*
aggregate SHUTDOWN_MESSAGE structure prefix OPC$ fill;
    SHUTDOWN_FILL       byte dimension OPC$K_HDR_SIZE fill;
    /*
    /* Define the minimum length, no special fields
    /*
    constant SHUTDOWN_MIN_SIZE equals .;        /* Min message size
    end SHUTDOWN_MESSAGE;
aggregate SHUTDOWN_OPTIONS structure longword unsigned prefix OPC$ fill;
    /*
    /* Define request dependent option bits.
    /*
    CLUSTER     bitfield mask;
    end SHUTDOWN_OPTIONS;

/*
/* Define CANCEL message fields
/*
aggregate CANCEL_MESSAGE structure prefix OPC$ fill;
    CANCEL_FILL byte dimension OPC$K_HDR_SIZE fill;
    /*
    /* Define the minimum length, no special fields
    /*
    constant CANCEL_MIN_SIZE equals .;  /* Min message size
    end CANCEL_MESSAGE;
```

```
aggregate CANCEL_OPTIONS structure longword unsigned prefix OPC$ fill;
    /*
    /* Define request dependent option bits.
    /*
    RQSTDONE    bitfield mask;
    end CANCEL_OPTIONS;

end_module OPCDEFTMP;
```

NPARSE
LIS

REQUEST
MAP

OPCOM

REPLY
MAP

NMLXFER
LIS

OPCCRASH
MAP

NMLV2STA
LIS

OPCOMLIB
B32

ALLOCATE
LIS

NMLZERO
LIS

OPCOMDEF
SDL

OPCOM
MAP

OPCDEFTMP
SDL

CANCEL
LIS