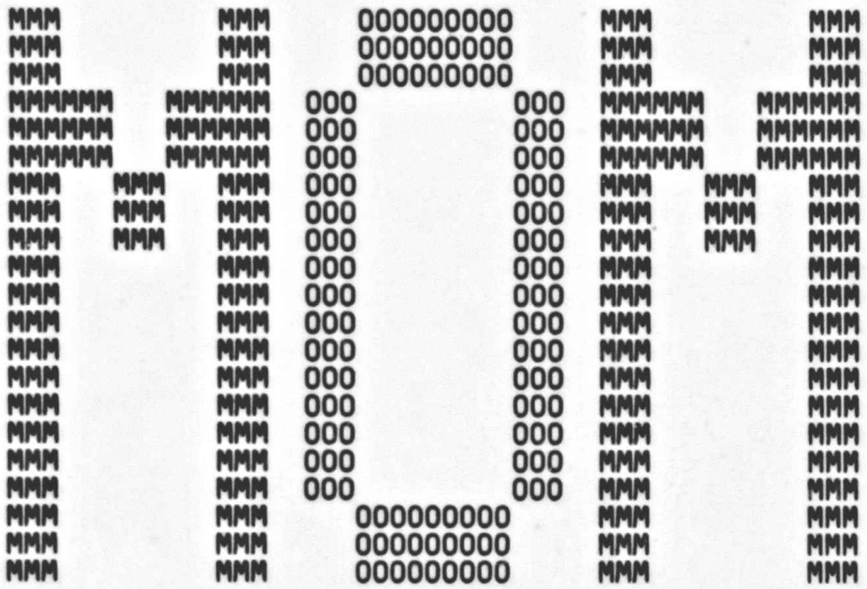
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMMMMM MMMMMM 000 000 MMMMMM MMMMMM
MMMMMM MMMMMM 000 000 MMMMMM MMMMMM
MMMMMM MMMMMM 000 000 MMMMMM MMMMMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM MMM 000 000 MMM MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000 000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM
MMM MMM 000000000 MMM MMM

FILEID**MOMDEF

```
MM      MM    OOOOOO    MM      MM  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
MM      MM    OOOOOO    MM      MM  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
MMMM  MMMM  OO      OO  MMMM  MMMM  DD      DD  EE          FF
MMMM  MMMM  OO      OO  MMMM  MMMM  DD      DD  EE          FF
MM  MM  MM  OO      OO  MM  MM  MM  DD      DD  EE          FF
MM  MM  MM  OO      OO  MM  MM  MM  DD      DD  EE          FF
MM      MM  OO      OO  MM      MM  DD      DD  EEEEEEEE    FFFFFFFF
MM      MM  OO      OO  MM      MM  DD      DD  EEEEEEEE    FFFFFFFF
MM      MM  OO      OO  MM      MM  DD      DD  EE          FF
MM      MM  OO      OO  MM      MM  DD      DD  EE          FF
MM      MM  OO      OO  MM      MM  DD      DD  EE          FF          ....
MM      MM  OO      OO  MM      MM  DD      DD  EE          FF          ....
MM      MM    OOOOOO    MM      MM  DDDDDDDD    EEEEEEEEEE  FF          ....
MM      MM    OOOOOO    MM      MM  DDDDDDDD    EEEEEEEEEE  FF          ....
```

```
  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
```

```
{ MOMDEF.MDL - internal service definitions for Maintenance Operations Module
{
{ 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:  VAX/VMS DECnet Maintenance Operations Module (MOM)
{
{ ABSTRACT:
{
{       This file contains varions service SDL definitions for MOM.
{
{ ENVIRONMENT:
{
{       n/a
{
{ AUTHOR:  Kathy Perko          17-Dec-1982
{
{ MODIFICATION HISTORY:
{       V03-007 MKP0007         Kathy Perko     21-July-1984
{               Increase size of maximum MOP message buffers so they
{               can hold a fairly large loop message.  Other logic will
{               be added to recover gracefully if the loop message is
{               larger than this buffer.
{
{       V03-006 MKP0006         Kathy Perko     30-May-1984
{               Remove echo parameter from CIB P2 buffer so QNAs will do
{               service functions.
{
{       V03-005 MKP0005         Kathy Perko     12-April-1984
{               Add dump complete MOP function code.
{
{       V03-004 MKP0004         Kathy Perko     11-April-1984
```

```
{               Convert file to SDL.
{
{       V03-003 MKP0003         Kathy Perko     26-Mar-1984
{               Fix area 1 problem.
{
{       V03-002 MKP0002         Kathy Perko     11-Feb-1984
{               Use Remote Console protocol type for Boot message on NI
{               (instead of Load Dump protocol - the DEUNA doesn't care,
{               but the QNA does).
{
{       V03-001 MKP0001         Kathy Perko     10-May-1983
{               Allow load file names of up to 128 characters.
{
{--
```

```
{
{ Symbol definitions for the Network Management Maintenance Operations Module
{

module $MOMDEF;
/*
/* Internal codes used to identify entity type of current circuit,
/* line  or node.
/*
constant(
        NODE
      , NODEBYNAME
      , CIRCUIT
      , LINE
      ) equals 0 increment 1 prefix MOM tag $C;

/*
/* MOM constants used in parsing NICE messages.
/*
constant(
        NODE_ID_PARAM                                   /* Parameter is a node address and/or name
      , NODE_ADDR_PARAM                                 /* Parameter is a node address
      ) equals 0 increment 1 prefix MOM tag $C;

/*
/* Flags set in MOM$GL_SERVICE_FLAGS
/*

aggregate MOMDEF  union fill prefix MOM$;
    MOMDEF_BITS structure fill;
        AUTOSERVICE bitfield mask;                      /* Autoservice function in progress (as
                                                        /*       opposed to operator requested
                                                        /*       service function.
        NI_CIRC bitfield mask;                          /* Service function is on an Ethernet circuit.
        MOP_RCV_OUT bitfield mask;                      /* There is a MOP receive outstanding.
        LOOP_W_ASSIST bitfield mask;                    /* Loop an NI circuit using a third node to
                                                        /*       assist.
        LOOP_W_ACCESS_CTL bitfield mask;                /* Loop a node with using an access control
                                                        /*       string (supplied in the NICE message).
        NI_MULTICAST bitfield mask;                     /* Autoservice request received on an NI was
                                                        /*       a multicast.
        CONSOLE_CARRIER_LOAD bitfield mask;             /* Down line loading console carrier code
                                                        /*       (needed because the file format is
                                                        /*       different than other load files).
        NI_VOLUNTEERING bitfield mask;                  /* A multicast service request was received
                                                        /*       from a target on the NI.  MOM is
                                                        /*       volunteering to perform the request.
                                                        /*       If the target does not respond, assume
                                                        /*       some other host was chosen by the target.
    end MOMDEF_BITS;
/*
/* MOM facility code
/*
    constant FAC_CODE   equals 519  prefix MOM tag $K;  /* Facility code
    constant SIG_CODE   equals 519*65536  prefix MOM tag $K;/* Signal code (519^16)
/*
```

```
/* MOM constants - QIO buffer sizes, etc.
/*
    constant NI_ADDR_LENGTH     equals 6  prefix MOM tag $K;/* Length of Ethernet addresses.
    constant NICE_BUF_LEN       equals 197 prefix MOM tag $K;/* NICE message buffer length
    constant QIO_BUF_LEN        equals 512  prefix MOM tag $K;/* QIO buffer length
    constant P2_BUF_LEN equals 104  prefix MOM tag $K;  /* Max length for P2 buffers.
    constant MAX_MOP_MSG_LEN    equals 1500 prefix MOM tag $K;/* Maximum length for a received MOP message
    constant NI_PREFIX  equals %X000400AA  prefix MOM tag $K;/* Standard NI prefix used for DEC Phase IV
                                                        /*       routers.
    constant LOADUMP_NI_PROT    equals %X0160  prefix MOM tag $K;/* NI Load/Dump Protocol type for NI driver
    constant CONSOLE_NI_PROT    equals %X0260  prefix MOM tag $K;/* NI Remote Console Protocol type.
    constant LOOP_NI_PROT       equals %X0090  prefix MOM tag $K;/* NI Loop Protocol type for NI driver
    constant MAX_LOOP_HEADER    equals 30  prefix MOM tag $K;/* Maximum size the NI loop message headers
                                                        /*       can be.
    constant NO_LOOP_HELP       equals -1  prefix MOM tag $K;/* Default value for circuit loop help type.
    constant SKIP_MULTICASTS    equals -1  prefix MOM tag $K;/* When looking for a MOP message response,
                                                        /*       skip over multicasts.
/*
/* Internal loopback type codes.
/*
    constant(
          LOOP_MOP                                      /* MOP loopback operation
        , LOOP_PHASE3                                   /* Phase III loopback operation
        , LOOP_PHASE2                                   /* Phase II loopback operation
        ) equals 0 increment 1 prefix MOM tag $C;

end MOMDEF;

end_module $MOMDEF;

module $MOPDEF;
```

```
/*
/* MOP message definitions
/*
                                                        /*
                                                        /* Define MOP function codes.
                                                        /*
constant FCT_MLT        equals 0  prefix MOP tag $;     /* Load/dump Memory load with transfer address
constant FCT_DCM        equals 1  prefix MOP tag $;     /* Dump Complete
constant FCT_MLD        equals 2  prefix MOP tag $;     /* Load/dump Memory load
constant FCT_ASV        equals 3  prefix MOP tag $;     /* Load/dump Assistance volunteer (NI only)
constant FCT_RMD        equals 4  prefix MOP tag $;     /* Request memory dump
constant FCT_RID        equals 5  prefix MOP tag $;     /* Remote Console Request ID
constant FCT_EMM        equals 6  prefix MOP tag $;     /* Remote Console Enter MOP mode (boot)
constant FCT_SID        equals 7  prefix MOP tag $;     /* Remote Console System ID
constant FCT_RPR        equals 8  prefix MOP tag $;     /* Load/dump Request program
constant FCT_RML        equals 10  prefix MOP tag $;    /* Load/dump Request memory load
constant FCT_RDS        equals 12  prefix MOP tag $;    /* Load/dump Request Dump Service (old MOP Mode Running)
constant FCT_MDD        equals 14  prefix MOP tag $;    /* Load/dump Memory dump data
constant FCT_PLT        equals 20  prefix MOP tag $;    /* Load/dump Parameter load with transfer address
constant FCT_ALD        equals 24  prefix MOP tag $;    /* Active loop data
constant FCT_PLD        equals 25  prefix MOP tag $;    /* Passive looped data
                                                        /*
                                                        /* Define MOP parameter codes.
                                                        /*
constant PAR_NNA        equals 1  prefix MOP tag $C;    /* Target node name
constant PAR_NAD        equals 2  prefix MOP tag $C;    /* Target node address
constant PAR_HNA        equals 3  prefix MOP tag $C;    /* Host node name
constant PAR_HAD        equals 4  prefix MOP tag $C;    /* Host node address
constant PAR_HTI        equals 5  prefix MOP tag $C;    /* Host system time

                                                        /*
                                                        /* Define MOP Software ID codes for Request Program and Boot messages.
                                                        /*
constant SID_NON        equals 0  prefix MOP tag $C;    /* No software ID
constant SID_OSY        equals %XFF  prefix MOP tag $C; /* Standard Operating System
constant SID_MAI        equals %XFE  prefix MOP tag $C; /* Maintenance System

                                                        /*
                                                        /* Define MOP Processor codes for Boot messages.
                                                        /*
constant PRO_SYS        equals 0  prefix MOP tag $C;    /* System processor
constant PRO_COM        equals 1  prefix MOP tag $C;    /* Communications processor (for loading
                                                        /*        Console Carrier)

                                                        /*
                                                        /* Define function codes used in NI loop messages.
                                                        /*
constant NILOOP_REPLY   equals 1  prefix MOP tag $C;    /* Reply to looped message
constant NILOOP_FORWARD equals 2  prefix MOP tag $C;    /* Forward looping message

end_module $MOPDEF;

module $SVDDEF;
```

```
/*
/*          Service Data
/* Definitions of target node service information.  This information is
/* retrieved from the volatile database and, if applicable, the NICE command.
/* It is saved in this table for use by Service during the service operation.
/*

aggregate SVDDEF structure fill prefix SVD$;
    NFB_ID_OVERLAY union fill;
        NFB_ID longword unsigned;                           /* NFB Field ID for QIOs to NETACP
        NFB_ID_FIELDS structure fill;
            FILL_1 byte dimension 3 fill prefix SVDDEF tag $$;
            NFB_DATABASE byte unsigned;                     /* NFB Field ID database (subfield of NFB_ID)
        end NFB_ID_FIELDS;
    end NFB_ID_OVERLAY;
    NICE_ID word unsigned;                                  /* NICE message parameter ID
    NICE_TYPE byte unsigned;                                /* Parameter type in NICE messages
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                                /* Flags field
        FLAGS_BITS structure fill;
            MSG_PARAM bitfield mask;                        /* The parameter value was supplied by the
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    STRING_LEN byte unsigned;                               /* Byte length of parameter value
    STRING_OVERLAY union fill;
        STRING character length 128;                        /* If it's a string parameter, the string.
        constant ENTRY_LEN equals . prefix SVD$ tag K;      /* Length of SVD table entries
        constant ENTRY_LEN equals . prefix SVD$ tag C;      /* Length of SVD table entries
        PARAM longword unsigned;                            /* If it's not a string parameter, the parameter
                                                            /*          value.

/*
/* Parametr type definitions
/*
        constant(
                'BYTE'
              , 'WORD'
              , LONG
              , STRING
              ) equals 0 increment 1 prefix SVD tag $K;


    end STRING_OVERLAY;
end SVDDEF;

end_module $SVDDEF;

module $MDTDEF;
```

```
/*
/* Mop Device Table definitions
/* This table contains the ASCII device name strings associated with a
/* given MOP device code.  It is used to construct secondary and tertiary
/* load file names when none are specified in the database.  The characters
/* 'SEC' or 'TER' are concatenated with the device name string to construct
/* a load file name.
/*


aggregate MDTDEF structure fill prefix MDT$;
    DEVTYPE byte unsigned;
    DEVSTRING address;
    constant ENTRYLEN equals . prefix MDT$ tag K;
    constant ENTRYLEN equals . prefix MDT$ tag C;

end MDTDEF;

end_module $MDTDEF;

module $CIBDEF;
```

```
/*
/* Channel Information Block (CIB)
/*

aggregate CIBDEF structure fill prefix CIB$;
    CHAN longword unsigned;
    NI_HIORD_ADDR_OVERLAY union fill;
        NI_HIORD_ADDR character length 6;                   /* HIORD NI address of target
        NI_HIORD_ADDR_FIELDS structure fill;
            NI_HIORD_PREF longword unsigned;                /* HIORD NI prefix assigned to Digital
            NI_HIORD_NODE word unsigned;                    /* Node address withing HIORD NI address of target
        end NI_HIORD_ADDR_FIELDS;
    end NI_HIORD_ADDR_OVERLAY;
    NI_HARDWR_ADDR character length 6;                      /* Hardware NI address of target
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                                /* Flags field
        FLAGS_BITS structure fill;
            TARGET_ADDR_FIXED bitfield mask;                /* Set when there is no alternate address
                                                            /*       to try in order to communicate with
                                                            /*       the target.  Applies to any service
                                                            /*       circuit, but is most useful on the NI
                                                            /*       where the target could be responding to
                                                            /*       either a hardware address or a DECnet
                                                            /*       physical address.
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    RETRY_CNT longword unsigned;                            /* Transmit retry count for transaction.
    SETMODE_P2_BUF_OVERLAY union fill;
        SETMODE_P2_BUF character length 54;                 /* P2 buffer for telling NI driver which NI
        constant CIBLEN equals . prefix CIB$ tag K;
        constant CIBLEN equals . prefix CIB$ tag C;
                                                            /*       address and protocol to use (among
                                                            /*       other things).
        SETMODE_P2_BUF_FIELDS structure fill;
            FILL_1 byte dimension 2 fill prefix CIBDEF tag $$;
            P2_BUF_SIZ longword unsigned;                   /* MOP message buffer size
            FILL_2 byte dimension 2 fill prefix CIBDEF tag $$;
            P2_PADDING longword unsigned;                   /* Put pad count on NI messages.
            FILL_3 byte dimension 14 fill prefix CIBDEF tag $$;
            P2_PROTOCOL longword unsigned;                  /* NI Protocol to use.
            FILL_4 byte dimension 18 fill prefix CIBDEF tag $$;
            NI_PHYS_ADDR character;                         /* Target NI physical address
        end SETMODE_P2_BUF_FIELDS;
    end SETMODE_P2_BUF_OVERLAY;
end CIBDEF;

end_module $CIBDEF;

module $NIHDEF;
```

```
/*
/* NI header definitions.
/* The NI device driver (XEDRIVER) returns the NI header for a message
/* in the P3 parameter.  There are several cases in which MOM needs to
/* get this header in order to find out which target on the a message
/* came from or whether or not it was sent to a multicast address.
/*


aggregate NIHDEF structure fill prefix NIH$;
    DEST_NI_ADDR_OVERLAY union fill;
        DEST_NI_ADDR character length 6;                /* destination Ni address of message
        MULTICAST byte unsigned;                        /* Low bit set if destination is a multicast
    end DEST_NI_ADDR_OVERLAY;
    SOURCE_NI_ADDR character length 6;                  /* Source Ni address of message
    PROTOCOL_TYPE word unsigned;                        /* Protocol type of message
    constant NI_HEADER_LEN equals . prefix NIH$ tag K;
    constant NI_HEADER_LEN equals . prefix NIH$ tag C;

end NIHDEF;

end_module $NIHDEF;

module $MSBDEF;
```

```
/*
/* Message segment block (MSB) definitions.  The MSB is used to accumulate
/* information with which to build the NICE response message returned to NCP
/* when the operation is completed.
/*


aggregate MSBDEF structure fill prefix MSB$;
    FLAGS_OVERLAY union fill;
        FLAGS longword unsigned;                        /* Flags
        FLAGS_BITS structure fill;
            CODE_FLD bitfield mask;                     /* Status code present (not used)
            DET_FLD bitfield mask;                      /* Error detail field present (DETAIL)
            MSG_FLD bitfield mask;                      /* Message text field present (TEXT)
            MSG2_FLD bitfield mask;                     /* Second line of message text present (TEXT2)
            ENTD_FLD bitfield mask;                     /* Enitity descriptor field present (ENTITY)
            DATA_FLD bitfield mask;                     /* Data descriptor field present (DATA)
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    FILL_1_OVERLAY union fill;
        FILL_1 longword fill prefix MSBDEF tag $$;
        CODE byte unsigned;                             /* Status code
    end FILL_1_OVERLAY;
    FILL_2_OVERLAY union fill;
        FILL_2 longword fill prefix MSBDEF tag $$;
        DETAIL word unsigned;                           /* Detail
    end FILL_2_OVERLAY;
    TEXT longword unsigned;                             /* Status code for text message.
    TEXT2 longword unsigned;                            /* Status code for second line of text msg.
    ENTITY address;                                     /* Entity descriptor address
    DATA address;                                       /* Data descriptor address
    constant "LENGTH" equals . prefix MSB$ tag K;       /* Maximum MSB size
    constant "LENGTH" equals . prefix MSB$ tag C;       /* Maximum MSB size
end MSBDEF;

end_module $MSBDEF;

module $DBGDEF;
```

```
/*
/* NML internal logging (debugging) flags
/*
/* These flags are used to enable logging of specified data to the NML log
/* file.  The flags are defined by translating the logical name NML$LOG, which
/* MOM uses also.  The permanent database and event logging flags are not
/* used by MOM.
/*

constant(
      NETIO                                         /* Network send/receive logging
    , FILEIO                                        /* File read/write logging
    , NPARSE                                        /* NPARSE state transition logging
    , LOOPIO                                        /* Loopback transmit/receive logging
    , ACPQIO                                        /* NETACP QIO logging
    , MOPIO                                         /* MOP send/receive logging - see MOP_xxx flags below.
    , SRVTRC                                        /* Trace service operations
    , EVENTS                                        /* Network event (EVL) logging

/*
/* The following flags must be set in conjunction with the DBG$C_MOPIO
/* flag.
/*
    , MOP_OTHER                                     /* All other MOP send/receive messages
    , MOP_MLD                                       /* MOP Memory Load Data messages (MOP$_FCT_MLD)
    , MOP_RML                                       /* MOP Request Memory Load messages (MOP$_FCT_RML)
    , MOP_RMD                                       /* MOP Request Memory Dump messages (MOP$_FCT_RMD)
    , MOP_MDD                                       /* MOP Memory Dump Data messages (MOP$_FCT_MDD)
    ) equals 0 increment 1 prefix DBG tag $C;

end_module $DBGDEF;
```

MOMBLDMSG
LIS

MOMINISTA
LIS

MOMLOAD
LIS

MOMACPIO
LIS

MOMDAT
LIS

NMAHEAD
B32

NPADEF
SDL

MOMDEF
SDL

NMATAIL
B32

MOMDDL
B32

NPAMAC
MAR

MOMLIB
LIS

MOMFILTO
LIS