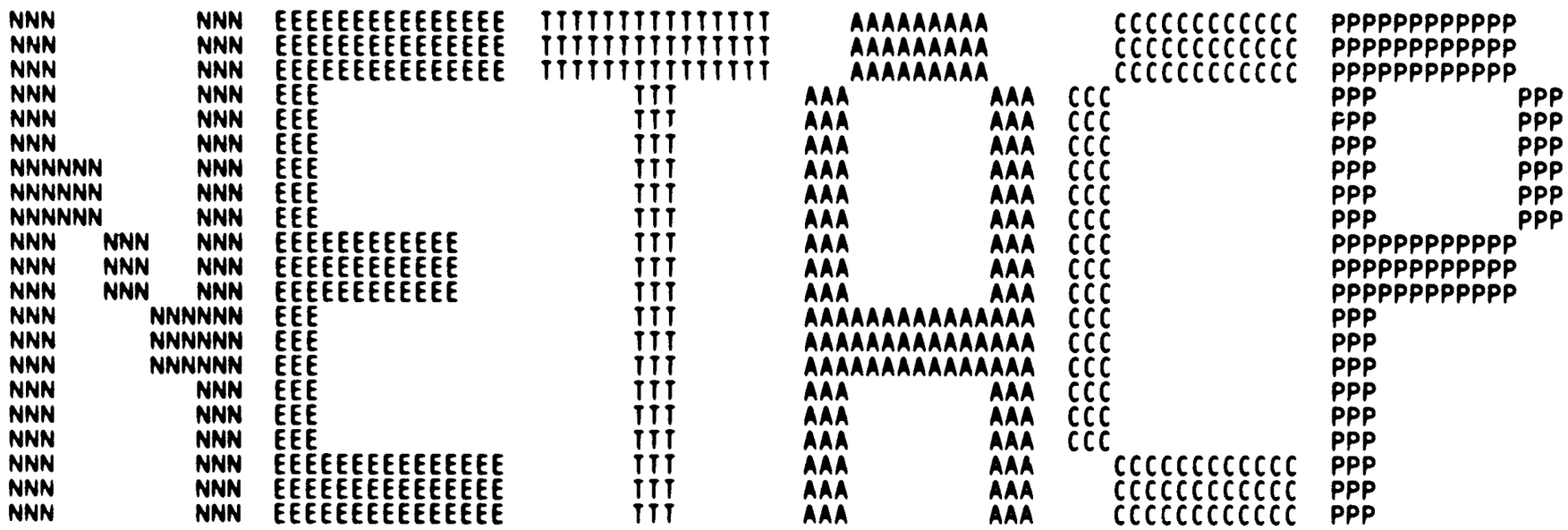
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP

FILEID**NETNPAGED

```
NN       NN  EEEEEEEEEE  TTTTTTTTTT  NN       NN  PPPPPPPP      AAAAAA      GGGGGGGG  EEEEEEEEEE  DDDDDDDD
NN       NN  EEEEEEEEEE  TTTTTTTTTT  NN       NN  PPPPPPPP      AAAAAA      GGGGGGGG  EEEEEEEEEE  DDDDDDDD
NN       NN  EE              TT      NN       NN  PP       PP  AA      AA  GG          EE          DD      DD
NN       NN  EE              TT      NN       NN  PP       PP  AA      AA  GG          EE          DD      DD
NNNN     NN  EE              TT      NNNN     NN  PP       PP  AA      AA  GG          EE          DD      DD
NNNN     NN  EE              TT      NNNN     NN  PP       PP  AA      AA  GG          EE          DD      DD
NN  NN   NN  EEEEEEEE        TT      NN  NN   NN  PPPPPPPP    AA      AA  GG          EEEEEEEE    DD      DD
NN  NN   NN  EEEEEEEE        TT      NN  NN   NN  PPPPPPPP    AA      AA  GG          EEEEEEEE    DD      DD
NN    NNNN  EE              TT      NN    NNNN  PP          AAAAAAAAAA  GG  GGGGGG  EE          DD      DD
NN    NNNN  EE              TT      NN    NNNN  PP          AAAAAAAAAA  GG  GGGGGG  EE          DD      DD
NN      NN  EE              TT      NN      NN  PP          AA      AA  GG      GG  EE          DD      DD      ....
NN      NN  EE              TT      NN      NN  PP          AA      AA  GG      GG  EE          DD      DD      ....
NN      NN  EEEEEEEEEE      TT      NN      NN  PP          AA      AA    GGGGGG    EEEEEEEEEE  DDDDDDDD        ....
NN      NN  EEEEEEEEEE      TT      NN      NN  PP          AA      AA    GGGGGG    EEEEEEEEEE  DDDDDDDD        ....

  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
```

```
{
{               NETNPAGED.SDL   - DECnet non-paged control 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.                  *
{*                                                                           *
{*****************************************************************************

{ MODIFIED BY:
{
{       V03-020 TMH0020         Tim Halvorsen   22-Apr-1984
{               Add ELECT_TIM flag to LPD for the election suppression timer.
{               Add PVC_ACCESSED flag to indicate PVC is ACCESSed.
{
{       V03-019 RNG0019         Rod Gamache     24-Mar-1984
{               Add NET$V_ACT bit to RCB$B_STATUS byte.
{
{       V03-018 PRB0319         Paul Beck       9-Mar-1984  9:34
{               Add NETUPD$_TEST_ADJ
{
{       V017    RNG0017         Rod Gamache     17-Feb-1984
{               Add NETUPD$_CRELNK for creating logical links.
{               Add ACT_TIMER and AQB_CNT bytes to the RCB.
{
{       V016    TMH0016         Tim Halvorsen   11-Jul-1983
{               Add executor cluster address to RCB.
{
{       V015    TMH0015         Tim Halvorsen   29-Mar-1983
{               Fix comments about MCOUNT offset
{
{       V014    TMH0014         Tim Halvorsen   13-Mar-1983
{               Add flag to disable AOA lookups (in the event we are
{               an isolated area router).
{               Add word/quadword datalink alignment flags.
{               Remove XMT_RESTR flag and add XMT_DALLY flag.
{
```

```
{       V013    TMH0013         Rod Gamache     25-Feb-1983
{               Add flag to toggle datalink line on circuit startup.
{
{       V012    TMH0012         Tim Halvorsen   14-Feb-1983
{               Add cell to hold datalink buffer size in LPD.
{               Remove XWB/LSB structures (now replaced by
{               a separate SDL).
{
{       V011    TMH0011         Tim Halvorsen   13-Oct-1982
{               Add storage for area routing support.
{
{       V010    TMH0010         Tim Halvorsen   28-Sep-1982
{               Add cell to RCB giving the designated router
{               if we are an endnode.
{
{       V009    RNG0001         Rod Gamache     20-Aug-1982
{               Add "broadcast to all endnodes" flag to LPD.
{
{       V008    TMH0008         Tim Halvorsen   07-Jul-1982
{               Add "broadcast circuit" flag to LPD.  Include "standard"
{               12 byte header.  Removed RCB$L_LOC_LPD and RCB$L_ECL_IRP.
{               Added LPD$C_LOC_INX.   Added RCB$L_PTR_JNX and RCB$L_PTR_DCS.
{               Add cost word to LPD.
{               Remove all fields in LPD, which are in new ADJ block.
{               Add RCB$W_MAX_RTG and MAX_ADJ.
{               Add LPD cell to point to a "most recently received election
{               message" for BC LPDs.
{               Add cell to RCB to hold the type of executor (routing,
{               endnode, etc.).
{               Add cell to LPD to hold the our NI router priority.
{               Add cell to LPD to hold the designated router on the circuit,
{               so that NETDRIVER can send the Router Hello msg to endnodes.
{               Add NETUPD message code to instruct NETDRIVER to send a hello
{               message immediately.
{               Add SRM bitmasks to LPD, in order to hold "send routing msg"
{               context for each circuit.  This is required for segmented
{               routing messages.  Remove RCB$V_STS_XRT flag, since that is
{               now replaced by the bitmasks.
{               Add cell to RCB to record current position in the ADJ vector
{               to be used by the listener process, since it will only do a
{               fraction of the adjacencies each second.
{               Add cell to LPD which determines the type of node we will
{               "look like" on that circuit (endnode, router, etc.)
{
{       V007    TMH0007         Tim Halvorsen   23-Jun-1982
{               Add CXB_FREE listhead to RCB
{
{       V006    TMH0006         Tim Halvorsen   19-May-1982
{               Add LPD$B_PVCFLG field and associated flags.
{               Add LPD$V_X25, LPD$V_X25BLK and LPD$V_INCOMING flags
{               Add LPD$B_STARTUPS to record number of startup
{               attempts since last "run" state
{               Add definitions for NI adjacency database.
{
{       V03-05  ADE0033         A.Eldridge      15-Feb-1982
{               Added RCB$B_ECL_RFLW and changed XWB$B_SPARE to XWB$B_INC_RFLW.
```

```
{
{       V03-04  ADE0032         A.Eldridge      15-Feb-1982
{               Added XWB$L_DI_CXB.
{
{       V03-03  ADE0031         A.Eldridge      18-Dec-1981
{               Added ICB$x_RID and XWB$x_RID.
{
{       V03-02  ADE0030         A.Eldridge      30-Nov-1981
{               Added RCB$B_ECL_DAC and RCB$B_ECL_DPX.
```

```
{
{ Routing Control Block definition.
{
{ The RCB is used as the network Volume Control Block.
{
{ Because miscellaneous software scans the I/O data base, the following fields
{ should be kept at the same offset as their VCB counter-parts:
{
{       RCB$W_TRANS     = VCB$W_TRANS    (offset = 12 decimal)
{       RCB$L_AQB       = VCB$L_AQB      (  ''    = 16    ''   )
{
module $RCBDEF;


aggregate RCBDEF structure fill prefix RCB$;
    IRP_FREE quadword unsigned;                 /* Listhead of free IRPs
    SIZE word unsigned;                         /* Bytes allocated for structure
    TYPE byte unsigned;                         /* Structure type
    STATUS_OVERLAY union fill;
        STATUS byte unsigned;                   /* Status flags
        STATUS_BITS structure fill;
            LVL2 bitfield mask;                 /* True if we can use level 2 routing (AOA, etc)
                                                /* False if we detect that we are an isolated area router
            ACT bitfield mask;                  /* (ROUTING LAYER BIT) Set if NETACP is considered active
        end STATUS_BITS;
    end STATUS_OVERLAY;
    TRANS word unsigned;                        /* Outstanding transaction count
    ADDR word unsigned;                         /* Local node address

    AQB longword unsigned;                      /* Ptr to AQB
    ACP_UCB longword unsigned;                  /* Ptr to Network ACP's UCB
    PTR_JNX longword unsigned;                  /* Ptr to journal buffer
    PTR_OA longword unsigned;                   /* Ptr to Output Adjacency vector
    PTR_AOA longword unsigned;                  /* Ptr to Area Output Adjacency vector

    PTR_LTB longword unsigned;                  /* Ptr to logical link table (dispatch vector)
    PTR_LPD longword unsigned;                  /* Ptr to Logical Path Descriptor vector
    PTR_ADJ longword unsigned;                  /* Pointer to vector of adjacency blocks (0 based)
    PTR_TQE longword unsigned;                  /* Ptr to internal Timer Queue element
    PTR_NDC longword unsigned;                  /* Ptr to Node Counters vector

    PTR_DCS longword unsigned;                  /* Ptr to NETDRIVER's control storage
    LOC_RCV quadword unsigned;                  /* Receive listhead for "local" Datalink
    LOC_XMT quadword unsigned;                  /* Transmit listhead for "local" Datalink
    IRP_WAIT quadword unsigned;                 /* Listhead of fork processes waiting for an IRP

    MCOUNT word unsigned;                       /* Mount count - includes current logical links plus 1
                                                /* for the ACP reference
    CUR_LNK word unsigned;                      /* Current number of logical links
    MAX_LNK word unsigned;                      /* Max allowable logical links
    MAX_ADDR word unsigned;                     /* Max allowable node address

    MAX_LPD byte unsigned;                      /* Max number of DLL LPDs
    MAX_SNK byte unsigned;                      /* Max IRPs queueable to any sink queue
    MAX_VISIT byte unsigned;                    /* Max nodes a packet may visit
    INT_PTH byte unsigned;                      /* Index of path to intercept node: 0=>no intercept
```

```
    ACT_DLL byte unsigned;                  /* Active data links
    STI byte unsigned;                      /* Internal state
    ECL_RFLW byte unsigned;                 /* ECL receiver pipeline quota (in packets)
    ECL_RFA byte unsigned;                  /* ECL retransmit factor

    ECL_DFA byte unsigned;                  /* ECL delay factor
    ECL_DWE byte unsigned;                  /* ECL delay weight
    ECL_DAC byte unsigned;                  /* ECL default access state
    ECL_DPX byte unsigned;                  /* ECL default proxy access state

    MAX_ADJ word unsigned;                  /* Size of adjacency vector (PTR_ADJ), in entries
    MAX_RTG word unsigned;                  /* Number of "routing destinations" (LPDs + BRAs)
    DLE_XWB longword unsigned;              /* Linked list head for direct-circuit-access XWBs

    TIM_RSI word unsigned;                  /* Min delay before next routing update in seconds
    TIM_RTI word unsigned;                  /* Max time before next routing update in seconds

    TIM_IAT word unsigned;                  /* Max logical link inactivity interval in seconds
    TIM_CNI word unsigned;                  /* Max inbound connect interval in seconds
    TIM_CNO word unsigned;                  /* Max outbound connect interval in seconds
    TIM_CTI word unsigned;                  /* Sample interval in seconds

    ECLSEGSIZ word unsigned;                /* Default ECL data segment size (excludes header size)
    TOTBUFSIZ word unsigned;                /* Buffer size including NSP header + 6 bytes
                                            /* + CXB$C_OVERHEAD bytes for datalink overhead
    CUR_PKT word unsigned;                  /* Current total IRPs
    MAX_PKT word unsigned;                  /* Max total IRPs

    PKT_FREE word unsigned;                 /* Packets left in Transmit pool
    PKT_PEAK word unsigned;                 /* Peak number of packets used

    PKT_FAIL byte unsigned;                 /* Packet allocation failures (over packet quota)
    MEM_FAIL byte unsigned;                 /* Packet allocation failures (insufficent memory)
    ETY byte unsigned;                      /* Type of executor node (ADJ$C_PTY_xxx)
    HOMEAREA byte unsigned;                 /* Our home area number (only if area router)

    MAX_AREA byte unsigned;                 /* Max allowable area address
    "ALIAS" word unsigned;                  /* Alias local address (0 if none)
    ACT_TIMER byte unsigned;                /* ACP activity timer
                                            /*
                                            /*        Transport layer counters
                                            /*
    ABS_TIM longword unsigned;              /* Absolute time counters were last zeroed
    CNT_NOL_OVERLAY union fill;
        CNT_NOL byte unsigned;              /* Node out-of-range packet loss
        CNT_1ST byte unsigned;              /* First counter cell marker
        constant CNT_SIZE       equals i2  prefix RCB tag $C;/* Number of bytes used for counters
    end CNT_NOL_OVERLAY;
    CNT_APL byte unsigned;                  /* Aged packet loss
    CNT_OPL byte unsigned;                  /* Oversized packet loss
    CNT_PFE byte unsigned;                  /* Packet format error

    CNT_RUL byte unsigned;                  /* Partial routing update loss
    CNT_VER byte unsigned;                  /* Verification rejects
    CNT_NUL word unsigned;                  /* Node unreachable packet loss
```

```
    CNT_XRE word unsigned;                          /* Xmitted connect resource errors (ECL layer counter)
    CNT_MLL word unsigned;                          /* Maximum logical links active (ECL layer counter)
                                                    /*
                                                    /*        End of transport counters
                                                    /*

    CXB_FREE quadword unsigned;                     /* Free CXB listhead
    LSN_ADJ byte unsigned;                          /* Current position in ADJ for listener processing
    AQB_CNT byte unsigned;                          /* Count of entries on AQB
    DRT word unsigned;                              /* Designated router ADJ index for all transmits
                                                    /* with an unspecified output adjacency
    LVL2 word unsigned;                             /* Level 2 router ADJ index for all transmits with
    constant "LENGTH" equals . prefix RCB$ tag K;   /* Structure size
    constant "LENGTH" equals . prefix RCB$ tag C;   /* Structure size
                                                    /* an unspecified output adjacency (ETY=P4 only)
end RCBDEF;

end_module $RCBDEF;

module $LPDDEF;
```

```
/*
/* Logical Path Descriptor  (LPD)
/*
/* The following control block describes a path to a data sink/source -- either
/* a datalink driver (e.g., XMDRIVER) or an end communications level (ECL)
/* driver (e.g., NETDRIVER).
/*

aggregate LPDDEF structure fill prefix LPD$;
    REQ_WAIT quadword unsigned;                 /* Listhead of fork processes waiting for a 'request'
                                                /* slot on the LPD sink queue.
    SIZE word unsigned;                         /* Structure size
    TYPE byte unsigned;                         /* Structure type
    STARTUPS byte unsigned;                     /* Number of datalink startup attempts since last "run"
    WIND longword unsigned;                     /* Driver context - WIND field image for IRPs
    UCB longword unsigned;                      /* Driver context - UCB address

    CHAN word unsigned;                         /* ACP channel to device
    TIM_TLK word unsigned;                      /* "Talker" timer
    INT_TLK word unsigned;                      /* "Talker" interval (used to init TIM_TLK)
    TSTCNT byte unsigned;                       /* Number of test messages left to send before entering
                                                /* the RUN state
    ASTCNT byte unsigned;                       /* Number of outstanding ASTs
    IRPCNT byte unsigned;                       /* Number of outstanding IRPs queued by NETDRIVER
    ETY byte unsigned;                          /* Our node type, on this circuit
    XMT_SRL byte unsigned;                      /* Output "square root limiter" value
    XMT_IPL byte unsigned;                      /* Output queue "input packet limiter"
    PTH_OVERLAY union fill;
        PTH word unsigned;                      /* Path ID
        PTH_FIELDS structure fill;
            PTH_INX byte unsigned;              /* Path index
            PTH_SEQ byte unsigned;              /* Path sequence
            constant LOC_INX    equals 1  prefix LPD tag $C;/* Index associate with the primary "local" LPD
                                                /*
                                                /*        Status on control info
                                                /*
        end PTH_FIELDS;
    end PTH_OVERLAY;
    STS_OVERLAY union fill;
        STS word unsigned;                      /* Status bits as follows:
        STS_BITS structure fill;
            ACTIVE bitfield mask;               /* Path is active
            STRTIM bitfield mask;               /* Set if restart supression timer is ticking
            DLE bitfield mask;                  /* Set if in use for physical line service
            ACCESS bitfield mask;               /* Set if LPD is being "accessed" by a server process
            RUN bitfield mask;                  /* Set if active and in use for normal data msgs
            XBF bitfield mask;                  /* Set if Xmitter uses buffered I/O
            RBF bitfield mask;                  /* Set if Receiver uses buffered I/O
            X25 bitfield mask;                  /* Set if X.25 datalink mapping used on this circuit
            X25BLK bitfield mask;               /* Set if blocking requested for X.25 datalink
            INCOMING bitfield mask;             /* Set if X.25 circuit waiting for incoming call
            BC bitfield mask;                   /* Set if circuit is a broadcast circuit (NI)
            XEND bitfield mask;                 /* Set if "send hello msg to all endnodes" is requested
            TOGGLE bitfield mask;               /* Set if listener timeout on DMC line (toggles line)
            ALIGNW bitfield mask;               /* Set if datalink requires word alignment
```

```
            ALIGNQ bitfield mask;                   /* Set if datalink requires quadword alignment
            ELECT_TIM bitfield mask;                /* Set if election suppression timer ticking
        end STS_BITS;
    end STS_OVERLAY;
    XMTFLG_OVERLAY union fill;
        XMTFLG byte unsigned;                       /* Xmit flags -- since a FFS is used to schedule message
                                                    /* transimission, the order of these flags are crucial
        XMTFLG_BITS structure fill;
            XMT_DALLY bitfield mask;                /* Dally before sending a start msg (must preceed STR)
            XMT_STR bitfield mask;                  /* Xmit a Transport start msg
            XMT_VRF bitfield mask;                  /* Xmit a Transport verification msg
            XMT_IDLE bitfield mask;                 /* Signals "no more Transport init messages to send"
            XMT_RT bitfield mask;                   /* Xmit a Transport routing message
            XMT_HEL bitfield mask;                  /* Xmit a Transport hello message
            XMT_ART bitfield mask;                  /* Xmit a Transport Area routing message
        end XMTFLG_BITS;
    end XMTFLG_OVERLAY;
    PVCFLG_OVERLAY union fill;
        PVCFLG byte unsigned;                       /* X.25 PVC startup flags -- since a FFS is used to
                                                    /* schedule it, the order of these flags are crucial
        PVCFLG_BITS structure fill;
            PVC_ACCESS bitfield mask;               /* Issue an IO$_ACCESS to establish the connection
            PVC_RESTRT bitfield mask;               /* Issue a "restart confirmation"
            PVC_RESET bitfield mask;                /* Issue a "reset" or "reset confirmation"
            FILL_2 bitfield length 4 mask;          /* (reserve low 4 bits for state, 3 more bits unused)
            PVC_ACCESSED bitfield mask;             /* True if PVC ACCESSed
        end PVCFLG_BITS;
    end PVCFLG_OVERLAY;
    STI byte unsigned;                              /* Internal state used by the ACP for initialization
    SUB_STA byte unsigned;                          /* Circuit sub-state
    PLVEC byte unsigned;                            /* Associate PLVEC index
    COST byte unsigned;                             /* Circuit cost
    BCPRI byte unsigned;                            /* Circuit NI router priority
    FILL_1 byte fill prefix LPDDEF tag $$;          /* spare
    DRT word unsigned;                              /* Designated router on NI
    RTR_LIST longword unsigned;                     /* For BC LPDs, address of "most recently received
                                                    /* election message" from Router Hello messages
                                                    /* (stored as an byte-counted string)
    RCV_IRP longword unsigned;                      /* Address of suspended receiver IRP
                                                    /*
                                                    /*        Transport layer counters
                                                    /*
    ABS_TIM longword unsigned;                      /* Absolute time counters were last zeroed
    CNT_APR_OVERLAY union fill;
        CNT_APR longword unsigned;                  /* Arriving packets received
        CNT_1ST byte unsigned;                      /* First counter cell marker
    end CNT_APR_OVERLAY;
    CNT_TPR longword unsigned;                      /* Transit packets received
    CNT_DPS longword unsigned;                      /* Departing packets sent
    CNT_TPS longword unsigned;                      /* Transit packets sent
    CNT_ACL word unsigned;                          /* Arriving congestion loss
    CNT_TCL word unsigned;                          /* Transit congestion loss
    CNT_LDN byte unsigned;                          /* Line down events
    CNT_IFL byte unsigned;                          /* Initialization failures
    constant CNT_SIZE   equals 22  prefix LPD tag $C;   /* Number of bytes used for counters
    BUFSIZ word unsigned;                           /* Datalink buffer size including Transport overhead
```

```
    SRM_POS byte unsigned;                          /* (variable route header depending on the datalink)
    SRM_LEFT byte unsigned;                         /* Current position in transmit XMT_SRM bitmask (bit !)
    ASRM_POS byte unsigned;                         /* Number of transmit XMT_SRM bits left to check/process
    ASRM_LEFT byte unsigned;                        /* Current position in transmit XMT_ASRM bitmask (bit !)
                                                    /* Number of transmit XMT_ASRM bits left to check/process
    SRM byte unsigned tag G dimension 4;            /*(32/8) ; "Send routing message" flags (one per segment)
    XMT_SRM byte unsigned tag G dimension 4;        /* Copy of SRM flags used for rtg msg transmission
    constant SRM_NODES  equals 32  prefix LPD tag $C;   /* Number of nodes per segment(bit)
    constant SRM_SHFT   equals 5  prefix LPD tag $C;    /* Bit shift for node<->bit!
    constant SRM_SIZE   equals 32  prefix LPD tag $C;   /* Number of bits in SRM bitmask (allows for 1024 nodes)

    ASRM byte unsigned tag G dimension 4;           /*(1/8); "Send area routing message" flags (one per segment)
    XMT_ASRM byte unsigned tag G dimension 4;       /* Copy of ASRM flags used for rtg msg transmission
    constant ASRM_AREAS equals 64  prefix LPD tag $C;   /* Number of areas per segment(bit)
    constant ASRM_SHFT  equals 6  prefix LPD tag $C;    /* Bit shift for area<->bit!
    constant ASRM_SIZE  equals 1  prefix LPD tag $C;    /* Number of bits in ASRM bitmask (allows for 64 areas)

    CACHE longword unsigned;                        /* Address of endnode cache storage (endnodes only)
    constant "LENGTH" equals . prefix LPD$ tag K;   /* Structure size
    constant "LENGTH" equals . prefix LPD$ tag C;   /* Structure size
end LPDDEF;

end_module $LPDDEF;

module $ADJDEF;
```

```
/*
/* Adjaceny Node Data Base Block (ADJ)
/*
/* This block describes the contents of the Adjacency Node Data Base. The
/* Adjacency Node Data Base is used in conjunction with the LPD data base
/* to describe the destination to any node in the network.
/*

aggregate ADJDEF structure fill prefix ADJ$;
    STS_OVERLAY union fill;
        STS byte unsigned;                              /* Status flags
        STS_BITS structure fill;
            INUSE bitfield mask;                        /* Adjacency block in use
            RUN bitfield mask;                          /* Adjacency is up (can transmit pkts over it)
            RTG bitfield mask;                          /* Partner is a routing node (PH3,PH4-1,PH4-2)
            LSN bitfield mask;                          /* Listen timer is ticking
        end STS_BITS;
    end STS_OVERLAY;
    PTYPE byte unsigned;                                /* Type of partner node (routing, non-routing, etc.)
    constant PTY_UNK    equals -1  prefix ADJ tag $C;   /* Node type is unknown
    constant PTY_PH3    equals 0  prefix ADJ tag $C;    /* Phase III full routing
    constant PTY_PH3N   equals 1  prefix ADJ tag $C;    /* Phase III non routing
    constant PTY_PH2    equals 2  prefix ADJ tag $C;    /* Phase II
    constant PTY_AREA   equals 3  prefix ADJ tag $C;    /* Phase IV area routing
    constant PTY_PH4    equals 4  prefix ADJ tag $C;    /* Phase IV routing
    constant PTY_PH4N   equals 5  prefix ADJ tag $C;    /* Phase IV non routing
    LPD_OVERLAY union fill;
        LPD word unsigned;                              /* Path ID
        LPD_FIELDS structure fill;
            LPD_INX byte unsigned;                      /* Path index
            LPD_SEQ byte unsigned;                      /* Path sequence
        end LPD_FIELDS;
    end LPD_OVERLAY;
    PNA word unsigned;                                  /* Partner node address
    BUFSIZ word unsigned;                               /* Neighbor's block size
    INT_LSN word unsigned;                              /* Listener interval (computed from neighbor's hello)
    TIM_LSN word unsigned;                              /* Listener timer (seconds left)
    BCPRI byte unsigned;                                /* Broadcast priority
    constant 'LENGTH' equals . prefix ADJ$ tag K;       /* Structure size
    constant 'LENGTH' equals . prefix ADJ$ tag C;       /* Structure size
end ADJDEF;

end_module $ADJDEF;

module $XMCDEF;
```

```
/*+
/*
/* DMC counter block - provides offsets for SHOW QIO return of counters
/*
/*-



aggregate XMCDEF structure fill prefix XMC$;
    XMTBYTCNT longword unsigned;                    /* No. of bytes transmitted
    RCVBYTCNT longword unsigned;                    /* No. of bytes received
    XMTMSGCNT word unsigned;                        /* No. of msgs transmitted
    RCVMSGCNT word unsigned;                        /* No. of msgs received
    RCVNOBUF word unsigned;                         /* No. of "no buffer" NAKS received
    RCVHDRBCC word unsigned;                        /* No. of "header BCC error" NAKS received
    RCVDATBCC word unsigned;                        /* No. of "data BCC error" NAKS received
    XMTNOBUF word unsigned;                         /* No. of "no buffer" NAKS xmitted
    XMTHDRBCC word unsigned;                        /* No. of "header BCC error" NAKS xmitted
    XMTDATBCC word unsigned;                        /* No. of "data BCC error" NAKS xmitted
    XMTREPS word unsigned;                          /* No. of REPS xmitted
    RCVREPS word unsigned;                          /* No. of REPS received
    constant "LENGTH" equals . prefix XMC$ tag K;   /* Length of counter block
    constant "LENGTH" equals . prefix XMC$ tag C;   /* Length of counter block
end XMCDEF;

end_module $XMCDEF;

module $LTBDEF;
```

```
/*
/* LINK TABLE        - LTB
/*
/* This structure is maintained by NSP (NETDRIVER).  It contains all
/* local 'end communications layer' parameters and a vector of logical
/* link slots.
/*

aggregate LTBDEF structure fill prefix LTB$;
    SLT_NXT longword unsigned;                  /* Pointer into the link slot vector of the
                                                /* slot candidate to be tried the next time
                                                /* a link slot needs to be allocated
    SLT_TOT word unsigned;                      /* Total slots in vector
    SLT_LMT word unsigned;                      /* Total useable slots in vector
    SIZE word unsigned;                         /* Size, in bytes, of this structure
    TYPE byte unsigned;                         /* Structure identifier
    SPARE byte unsigned;                        /* Reserved for future use
    XWB longword unsigned;                      /* XWB listhead
    SLOTS longword unsigned;                    /* The begining of the logical link slot vector.
    constant "LENGTH" equals . prefix LTB$ tag K;   /* Structure size
    constant "LENGTH" equals . prefix LTB$ tag C;   /* Structure size
                                                /* This is the first slot, not a pointer to it.
                                                /* Each slot in the vector is 4 bytes.  If the
                                                /* low bit is set then the slot is available and
                                                /* its sequence number (number of times used) is
                                                /* found in the high order word.  If the low bit
                                                /* is clear then the slot contains a pointer to
                                                /* a structure containing the link context and
                                                /* state information.
                                                /*
                                                /* This vector is terminated by a longword of all
                                                /* ones followed by a longword of all zeroes.
end LTBDEF;
end_module $LTBDEF;
module $ICBDEF;
```

```
/*
/* INTERNAL CONNECT BLOCK  - used to pass generic connect information
/*                           between the Network ACP and an End Communi-
/*                           cations Layer (ECL) driver (e.g. NETDRIVER)
/*

aggregate ICBDEF structure fill prefix ICB$;
    PATH word unsigned;                                 /* Path number (logical line) over which connect
                                                        /* was received or is to be transmitted.  It
                                                        /* contains a sequence number in the high byte.
                                                        /* A value of zero indicates no path specified.
    LOCLNK word unsigned;                               /* Local link address
    TIM_OCON word unsigned;                             /* Outbound connect timer, in seconds
    TIM_INACT word unsigned;                            /* Maximum inactivity timer, in seconds
    SIZE word unsigned;                                 /* Bytes allocated for structure
    TYPE byte unsigned;                                 /* Contains structure type code
    FILL_1 byte fill prefix ICBDEF tag $$;              /* (spare)
    RETRAN word unsigned;                               /* Maximum consecutive retransmissions before the
                                                        /* link is forcibly disconnected
    DLY_FACT word unsigned;                             /* Retransmission timer delay factor
    DLY_WGHT word unsigned;                             /* Retransmission timer delay weight
    SEGSIZ word unsigned;                               /* Maximum receive data segment size, in bytes,
                                                        /* excluding End Communications Layer protocol.
    constant LPRNAM     equals 20  prefix ICB tag $C;   /* Maximum length of counted process name
    LPRNAM byte unsigned;                               /* Count of bytes used in next field
    LPRNAM character length 19;                         /* Local process name
    constant RPRNAM     equals 20  prefix ICB tag $C;   /* Maximun length of counted process name
    RPRNAM byte unsigned;                               /* Count of bytes used in next field
    RPRNAM_OVERLAY union fill;
        RPRNAM character length 19;                     /* Remote process name
                                                        /*
                                                        /* Process names contain a type field in the
                                                        /* first byte, and have the following formats:
                                                        /*
                                                        /* <0> followed by 1 byte non-zero object type
                                                        /* <1> followed by an object type followed by
                                                        /*     a counted (16 text bytes,max) object name
                                                        /* <2> followed by an object type followed by
                                                        /*     2 byte group code, 2 byte user code, and
                                                        /*     a counted (12 text bytes,max) object name
                                                        /*
        RPRNAM_FIELDS structure fill;
            DSTFMT byte unsigned;                       /* Format byte of remote process name
            DSTOBJ byte unsigned;                       /* Object type byte of remote process name
            DSTDSC character;                           /* Destination descriptor part of remote proc name
            constant ACCESS     equals 64  prefix ICB tag $C;/* Maximum combined composite access string length
        end RPRNAM_FIELDS;
    end RPRNAM_OVERLAY;
    ACCESS byte unsigned;                               /* Count of bytes in following field
    ACCESS character length 63;                         /* Concatenated access control strings, with subfield
                                                        /* maximum lengths as follows (note that the sum of
                                                        /* the actual subfield lengths must be less than 63):
/*    F         USER,B                  /* Count of bytes used in next field
/*    F         USER,T,39          /* "User" field of account to be connected to
/*    F         PSW,B                   /* Count of bytes used in next field
/*    F         PSW,T,39          /* "Password" field of that account
```

```
/*   F        ACCT,B                  /* Count of bytes used in next field
/*   F        ACCT,T,39          /* "Account" field of that account

    DATA byte unsigned;                                 /* Count of bytes used in next field
    DATA character length 16;                           /* Optional connect data
    REMNOD word unsigned;                               /* Network address of partner
    FILL_2 word fill prefix ICBDEF tag $$;              /* Spare for alignment
    FILL_3 byte fill prefix ICBDEF tag $$;              /* Spare for alignment
    RID byte unsigned;                                  /* Remote user (process, task, etc.) i.d.
    RID character length 16;                            /*
    constant "LENGTH" equals . prefix ICB$ tag K;       /* Structure size
    constant "LENGTH" equals . prefix ICB$ tag C;       /* Structure size
    constant RID        equals 16  prefix ICB tag $C;   /* Max size of RID text field
end ICBDEF;

end_module $ICBDEF;

module $NETUPDDEF;
```

```
/**
/*
/* Function codes for XWB update routine (ACP communication routine)
/*
/*-

                                                                /* Code
constant(
      ABORT                                                     /* Breaking link
    , CONNECT                                                   /* Give NCB to task w/ declared name or object
    , EXIT                                                      /* Task is exiting
    , PROCRE                                                    /* Starting a new process
    , DLL_ON                                                    /* Datalink starting for normal use
    , DLL_DLE                                                   /* Datalink starting for service functions
    , CRELNK                                                    /* Create a logical link control block
    , ABOLNK                                                    /* Abort all links - network shutdown
    , DSCLNK                                                    /* Disconnect the specified link
    , BRDCST                                                    /* Broadcast mailbox message
    , REPLY                                                     /* Send mailbox message to a specific mailbox
    , REACT_RCV                                                 /* Reactivate datalink receiver
    , SEND_HELLO                                                /* Send hello message immediately
    , GET_ADJ                                                   /* Get output ADJ for a given node address
    , TEST_ADJ                                                  /* Test if given node is adjacent to endnode
    ) equals 1 increment 1  prefix NETUPD tag $;

end_module $NETUPDDEF;

module $NETMSGDEF;

/**
/* Define event codes used to pass blocks to the ACP
/*
/*-

                                                                /*
constant(
      UNK                                                       /* Unknown message
    , ILL                                                       /* Illegal message
    , TR                                                        /* Transport control message
    , IRP                                                       /* IRP from datalink which is shutting down
    , APL                                                       /* Aged packet
    , NUL                                                       /* Node unreachable packet
    , NOL                                                       /* Node out of range packet
    , PFE                                                       /* Packet format error
    , LSN                                                       /* Listner timer expired
    , OPL                                                       /* Oversized packet loss
    , CRD                                                       /* Circuit run down
    , ADJ                                                       /* Adjacency up
    ) equals 1 increment 1  prefix NETMSG tag $C;

end_module $NETMSGDEF;
```

NETACP

NDDRIVER
MAP

NETDRIVER
MAP

NETSERVER
MAP

DLEDEF
SDL

NETACP
MAP

NETNPAGED
SDL

NMALIBRY
LIS

NETCTL
SDL