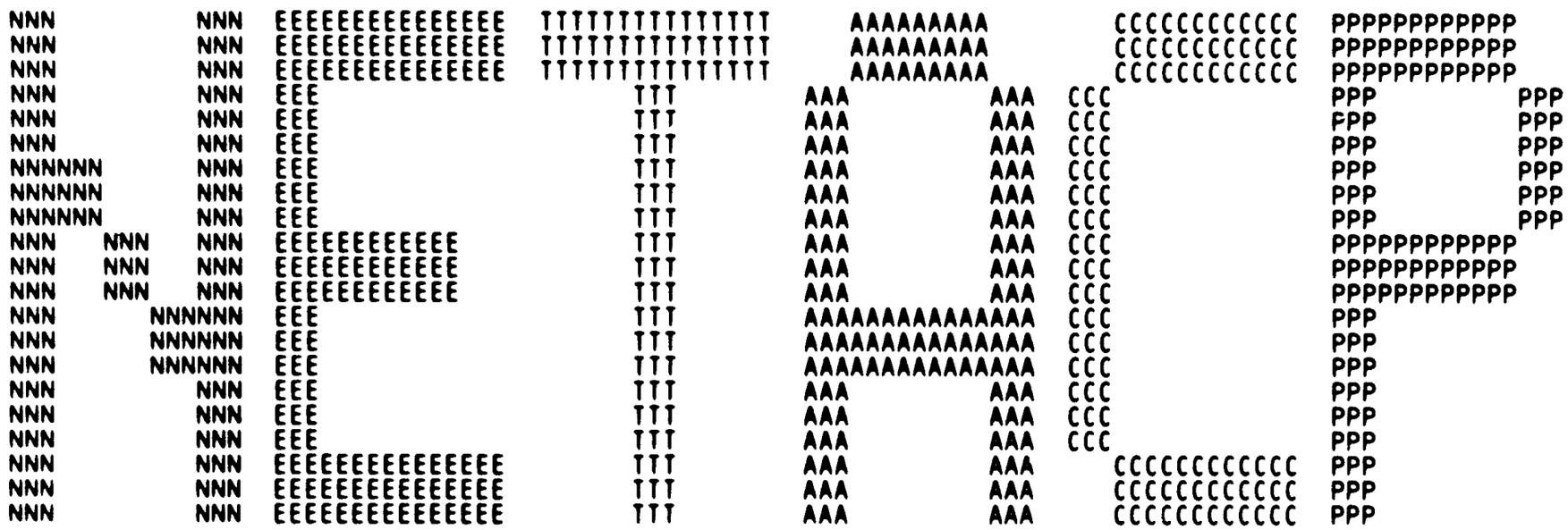
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN 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**XWBDEF

```
XX      XX  WW      WW  BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
XX      XX  WW      WW  BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
XX      XX  WW      WW  BB      BB  DD      DD  EE          FF
XX      XX  WW      WW  BB      BB  DD      DD  EE          FF
  XX  XX    WW      WW  BB      BB  DD      DD  EE          FF
  XX  XX    WW      WW  BB      BB  DD      DD  EE          FF
    XX      WW      WW  BBBBBBBB    DD      DD  EEEEEEEE    FFFFFFFF
    XX      WW      WW  BBBBBBBB    DD      DD  EEEEEEEE    FFFFFFFF
  XX  XX    WW  WW  WW  BB      BB  DD      DD  EE          FF
  XX  XX    WW  WW  WW  BB      BB  DD      DD  EE          FF
XX      XX  WWWW  WWWW  BB      BB  DD      DD  EE          FF              ....
XX      XX  WWWW  WWWW  BB      BB  DD      DD  EE          FF              ....
XX      XX  WW      WW  BBBBBBBB    DDDDDDDD    EEEEEEEEEE  FF              ....
XX      XX  WW      WW  BBBBBBBB    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
```

```
                MODULE  $xwbdef

{
{ Version:      'V04-000'
{
{*
{*              XWBDEF.SDL      - Logical-link nonpaged control structures
{*
{*****************************************************************************
{*                                                                           *
{*  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.                  *
{*                                                                           *
{*                                                                           *
{*****************************************************************************
{*
{*
{*  AUTHOR:             Alan D. Eldridge        1-April-1982
{*
{*  MODIFIED BY:
{*
{*      V03-006 RNG0006         Rod Gamache     11-Jun-1984
{*              Add XWB$V_STS_NDC node counter access bit to XWB.
{*
{*      V03-005 ADE0001         A.Eldridge      21-Oct-1982
{*              Significant LSB redefinition.  Removed DLE stuff from XWB.
{*
{*
```

```
{
{  LOGICAL-LINK SUBCHANNEL BLOCK   (LSB)
{
{  The block is used to control the activity on a logical link
{  subchannel.  There are two subchannels: the DATA subchannel and the
{  INTERRUPT/LINK SERVICE subchannel.
{
{
{
AGGREGATE  lsb STRUCTURE PREFIX lsb$ ;  { Link Subchannel Block

{   KEEP LONGWORDS, WORDS, AND BYTES ALIGNED ON THEIR APPROPRIATE BOUNDARIES

    {
    {  Transmitter control variables
    {

    lux         WORD;           { Last segment # assigned to a segment
    lnx         WORD;           { Last segment # xmitted
    hxs         WORD;           { Highest segment # sendable
    har         WORD;           { Highest ACK # received
                                {
    haa         WORD;           { Highest ACK # acceptable
    x_req       BYTE;           { Flow control credits from remote receiver
    x_adj       BYTE;           { Packet window adjustment counter
                                {
    x_pktwnd    BYTE;           { Size of the transmit-packet-window
    x_cxbact    BYTE;           { Number of active transmit CXB's
    x_cxbquo    BYTE;           { Max total CXB's allowed
    x_cxbcnt    BYTE;           { Total CXB's both active on on the free queue
                                {
    x_pnd       ADDRESS TAG L;  { Listhead for xmt IRP's containing data
    x_irp       ADDRESS TAG L;  { Listhead for xmt IRP's with data moved to CXBs
    x_cxb       ADDRESS TAG L;  { Transmit CXB (message segments) listhead

    {
    {  Receiver control variables
    {

    r_irp       ADDRESS TAG L;  { Receive IRP listhead
    r_cxb       ADDRESS TAG L;  { Received CXB (message segments) listhead
                                {
    hnr         WORD;           { Highest numbered msg received and accepted
    hax         WORD;           { Highest ACK xmitted
                                {
    r_cxbcnt    BYTE;           { Number of CXB's in LSB list (unACK'ed)
    r_cxbquo    BYTE;           { Max rcv CXB's which can be buffered by NSP
                                { before some are passed to the Session Layer
    spare       BYTE;           { Spare, used for alignment

    {
    {  Miscellaneous
    {

    sts         STRUCTURE TAG b;    { Status bits
```

```
        li      BITFIELD MASK;      { Set for LS/INT subchannel
        spare   BITFIELD LENGTH 4;  { skip over to coincide with NSP bits
        bom     BITFIELD MASK;      { Next seg to send has NSP$V_DATA_BOM set
        eom     BITFIELD MASK;      { Next seg to send has NSP$V_DATA_EOM set

    END sts;
cross       ADDRESS TAG L;  { Pointer to "cross-channel" LSB

#lsb_lng = .;               { Length for use by XWB definition
END lsb;                    {
```

```
{
{  NETWORK WINDOW BLOCK  (XWB)  - Network version of a WCB
{
{  This control block serves as the Network Window Control Block, as such
{  its header section must look like a WCB.  The remainder of the
{  structure is Network specific.  There is one XWB per logical link.
{
{

AGGREGATE  xwb STRUCTURE PREFIX xwb$ ;
    {
    {
    {  The header portion of the block tracks the WCB format
    {
    {
    wlfl        ADDRESS TAG L;  { Window list forward link
    wlbl        ADDRESS TAG L;  { Window list backward link
    size        WORD;           { Bytes allocated for structure
    type        BYTE;           { Contains code identifying structure type
    access      BYTE;           { IO$_ACCESS control flags (see WCB definition)
    refcnt      WORD;           { Count of accessors of the window
    sts         STRUCTURE TAG w;{ Contains the miscellaneous status flags.
                                { (DECnet specific)
        sts_tid     BITFIELD MASK;  { Set if XWB$W_TIM_ID is currently in use
        sts_tli     BITFIELD MASK;  { Set if XWB$W_TIM_ID used by LI subchannel
        sts_sol     BITFIELD MASK;  { Set if the XWB fork block is in use
        sts_dis     BITFIELD MASK;  { Set if synchronous disconnect is pending
                                    { awaiting RUN state run-down
        sts_con     BITFIELD MASK;  { Set until XWB enters the RUN state
        sts_tmo     BITFIELD MASK;  { Link timed out
        sts_rbp     BITFIELD MASK;  { Set if receiver is back-pressured off
        sts_ovf     BITFIELD MASK;  { Set if local rcvr is in "overflow" state
                                    {
    sts_dtnak   BITFIELD MASK;  { Set if next DATA ACK should be a NAK
    sts_linak   BITFIELD MASK;  { Set if next LS/INT ACK should be a NAK
    sts_astpnd  BITFIELD MASK;  { Special Kernel AST pending
    sts_astreq  BITFIELD MASK;  { Special Kernel AST requested
                                {
    sts_ndc     BITFIELD MASK;  { Set if NODE COUNTER BLOCK is accessed
                                {
    END sts;                    {
                                {
orgucb      ADDRESS TAG L;      { Original UCB address

{
{  The remainder of the block is DECnet specific.
{

fork        QUADWORD;           { Fork queue linkage
flg         STRUCTURE TAG w;    { Flags to control message xmission.
    {
    {   Because an FFS instruction is used on the following to determine
    {   what to do next, the order of the bit definitions is critical.
    {
    flg_break   BITFIELD MASK;  { Break the link
```

```
        flg_wbuf    BITFIELD MASK;  { Wait for buffer availability
        flg_siack   BITFIELD MASK;  { Send INT/LS ACK
        flg_sdack   BITFIELD MASK;  { Send DATA ACK
        flg_sli     BITFIELD MASK;  { Send INT/LS message
        {
        {   The next 2 bits are wait conditions on the DATA subchannel
        {
        flg_whgl    BITFIELD MASK;  { Wait for HXS to become less than LUX
        flg_wbp     BITFIELD MASK;  { Wait for backpressure to be relaxed
                                    {
        flg_sdt     BITFIELD MASK;  { Send a DATA message
        flg_scd     BITFIELD MASK;  { Send Connect or Disconnect
        flg_clo     BITFIELD MASK;  { Close the link and deallocate the XWB
        flg_wdat    BITFIELD MASK;  { Waiting for transmit CXB useage to go
                                    { below quota
        {
        {  The next 4 flags are used for controlling when and how the next
        {  Link Service subchannel message is to be built.  They do not by
        {  themselves indicate that the message is to be transmitted.
        {
        {  Since the are scanned via a FFS instruction, they must be defined
        {  in order of their priority.
        {
        flg_tbpr    BITFIELD MASK;  { Toggle back-pressure at remote end
        flg_iavl    BITFIELD MASK;  { Build Interrupt message
        flg_sifl    BITFIELD MASK;  { Build INT flow ctl msg
        flg_sdfl    BITFIELD MASK;  { Build DATA flow ctl msg
                                    {
        END flg;                    {
                                    {
    sta         BYTE;               { Logical link states
        CONSTANT  (sta_clo,         { Closed state
                   sta_cis,         { Connect Initiate Sending
                   sta_car,         { Connect Ack Received
                   sta_cir,         { Connect Initiate Received
                   sta_ccs,         { Connect Confirm Sending
                   sta_run,         { Run
                   sta_dir,         { Disconnect Initiate Received
                   sta_dis,         { Disconnect Initiate Sending
                   numsta           { Number of states (this works since
                  )                 {  these constants start at "0").  This
                EQUALS 0 INCREMENT 1{  value (8) allows a quadword per
                TAG c;              {  event in the state tables. Changing
                                    {  it may be difficult.
    fipl        BYTE;               { Fork IPL level
    fpc         ADDRESS TAG L;      { Fork process PC value
    fr3         LONGWORD;           { Fork process R3 value
    fr4         LONGWORD;           { Fork process R4 value
    link        ADDRESS TAG L;      { Link for XWB list
    vcb         ADDRESS TAG L;      { Ptr to Volume Control Block (actually RCB)
    pid         LONGWORD;           { I.D. of process given connect
    path        WORD;               { Path (circuit number) over which to xmit.
                                    { Zero implies choose the circuit number
                                    { dynamically.  High byte is a sequence #
    remnod      WORD;               { Network address of partner node
    remlnk      WORD;               { Remote node's link address
```

```
    loclnk      WORD;               { Local link address
    locsiz      WORD;               { Maximum receive segment size
    remsiz      WORD;               { Maximum transmit segment size
    r_reason    WORD;               { Received disconnect reason
    x_reason    WORD;               { Disconnect reason code to send to remote
    tim_id      WORD;               { Identity of segment being timed
    elapse      WORD;               { Seconds since the timer was last reset
    tim_inact   WORD;               { Maximum inactivity interval, in seconds
    delay       WORD;               { Estimated seconds between xmission and ACK
    timer       WORD;               { Current timer value, in seconds
    progress    WORD;               { Logical link confidence variable
    retran      WORD;               { Maximum retransmissions before link is to
                                    { be disconnected
    dly_fact    WORD;               { Retransmission timer delay factor
    dly_wght    WORD;               { Retransmission timer delay weight
    pro         STRUCTURE TAG b;    { Partner's protocol capabilities
        pro_nfc     BITFIELD MASK;  { Partner's receiver uses "no-flow"
        pro_sfc     BITFIELD MASK;  { Partner's receiver uses "segment flow"
        pro_ph2     BITFIELD MASK;  { Parnter is Phase II (no timer support)
        pro_cca     BITFIELD MASK;  { Cross-channel ACKing allowed
        pro_nar     BITFIELD MASK;  { Can send "no ack request" flag to partner
                                    {
        END pro;                    {
                                    {
    #x = 16;                        { Size DATA field
    CONSTANT  data                  { Max size of DATA text field
                EQUALS #x TAG c;    {
    data        BYTE;               { Count of bytes used in next field
    data        CHARACTER           { Optional data to be sent in next connect
                LENGTH #x;          { or disconnect message
    x_flw       BYTE;               { Transmit link service/flow control info
    x_flwcnt    BYTE;               { Flow control count for next link service
                                    { message to be transmitted
    sp3         BYTE;               { Spare for alignment
    #x = 16;                        { Size RID field
    CONSTANT  rid                   { Max size of RID text field
                EQUALS #x TAG c;    {
    rid         BYTE;               { Remote user (process, task, etc.) i.d.
    rid         CHARACTER           {
                LENGTH #x;          {
    irp_acc     LONGWORD;           { Ptr to IO$_ACCESS IRP (0 if none)
    #ndc = 32;                      {
    CONSTANT  ndc_lng               { Define room for counter block
                EQUALS #ndc TAG c;  {
    ndc         BYTE TAG  z         { Counter block area
                DIMENSION #ndc ;
    {
    {
    {  The remainder of the structure is multiplexed depending upon the NSP
    {  logical-link state, and depending upon whether the XWB is used for
    {  logical-links or for Direct-line access.
    {
    {
    comlng      UNION TAG c;        { XWB$C_COMLNG is the length of the common
                                    { XWB area above.
      run_blk   STRUCTURE;          { While in the RUN state
```

```
        ''      CHARACTER LENGTH ((.+3)&(-4))-.;{ Force longword alignment
        dt      CHARACTER LENGTH #lsb_lng;      { DATA subchannel LSB
        ''      CHARACTER LENGTH ((.+3)&(-4))-.;{ Force longword alignment
        li      CHARACTER LENGTH #lsb_lng;      { LS/INT subchannel LSB
        dea_irp LONGWORD;                       { IO$_DEACCESS IRP
        END     run_blk;                        {
                                                {
      con_blk   STRUCTURE;                      { For all "connect" states the field usage
                                                { is as follows. Process names contain a
                                                { format field in the first byte, and have
                                                { the following formats:
                                                {
                                                { 0  followed by 1 byte non-zero object #
                                                { 1  followed by 1 byte of zero (object #)
                                                {    followed by a counted (max 16 bytes of
                                                {    test) object name
                                                { 2  followed by 1 byte of zero (object #)
                                                {    followed by 2 byte group code, 2 byte
                                                {    user code, and a counted (max 12 bytes
                                                {    of text object name
                                                {
        #y = 19;                                { Max size of process name field
        CONSTANT (lprnam,rprnam)                { Size of the following count and text
                EQUALS #y+1 TAG c;              { fields
        lprnam  BYTE;                           { Count field for local process name
        lprnam  CHARACTER LENGTH #y;            { Local process name
        rprnam  BYTE;                           { Count field for remote process name
        rprnam  CHARACTER LENGTH #y;            { Remote process name
                                                {
        #y = 63;                                { Max size of LOGIN compound string
        CONSTANT login EQUALS #y+1              { Size of LOGIN compound strings + count
                        TAG c;                  { field
        login   BYTE;                           { Count of the following composite field
                                                {  (mininum value is 3)
        login   CHARACTER LENGTH #y;            { Three concatenated counted account strings
                                                { (min value 3 consecutive 0's)
        icb     ADDRESS TAG l;                  { Pointer to Internal Connect Block
        ci_path WORD;                           { Path i.d. over which Connect Initiate
                                                { was received.
        #y = .;                                 { Length to this point
        CONSTANT conlng EQUALS #y               {
                        TAG c;                  {
        END con_blk;                            {
      END conlng;                               {

      ''  CHARACTER LENGTH ((.+7)&(-8))-.; { Force quadword alignment
      free_cxb    QUADWORD;                 { Listhead of free CXB's
                                            {
      END xwb;                              {


END_MODULE  $xwbdef;
```

NETUSR
SDL

LIBTAIL
B32

XWBDEF
SDL

NETDEFS
MAR

PSIUSR
SDL

NETDRVMAC
MAR

NODRIVER
LIS

NSPMSGDEF
SDL

LIBHEAD
B32

NET
LIS

NETMACROS
MAR

NETACPTRN
LIS