

|        |        |           |           |         |         |         |         |
|--------|--------|-----------|-----------|---------|---------|---------|---------|
| NNN    | NNN    | EEEEEEEEE | TTTTTTTTT | AAAAAAA | CCCCCCC | PPPPPPP |         |
| NNN    | NNN    | EEEEEEEEE | TTTTTTTTT | AAAAAAA | CCCCCCC | PPPPPPP |         |
| NNN    | NNN    | EEEEEEEEE | TTTTTTTTT | AAAAAAA | CCCCCCC | PPPPPPP |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNNNNN | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNNNNN | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNNNNN | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNNNNN | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | NNN       | EEEEEEEEE | TTT     | AAA     | CCC     | PPPPPPP |
| NNN    | NNN    | NNN       | EEEEEEEEE | TTT     | AAA     | CCC     | PPPPPPP |
| NNN    | NNN    | NNN       | EEEEEEEEE | TTT     | AAA     | CCC     | PPPPPPP |
| NNN    | NNNNNN | EEE       | TTT       | AAAAAAA | CCC     | PPP     |         |
| NNN    | NNNNNN | EEE       | TTT       | AAAAAAA | CCC     | PPP     |         |
| NNN    | NNNNNN | EEE       | TTT       | AAAAAAA | CCC     | PPP     |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | EEE       | TTT       | AAA     | CCC     | PPP     |         |
| NNN    | NNN    | EEEEEEEEE | TTT       | AAA     | CCCCCCC | PPP     |         |
| NNN    | NNN    | EEEEEEEEE | TTT       | AAA     | CCCCCCC | PPP     |         |
| NNN    | NNN    | EEEEEEEEE | TTT       | AAA     | CCCCCCC | PPP     |         |

NE

NE

SR

S  
Ps  
--  
NE

\*\*FILE\*\*ID\*\*NETUSR

B 1

NN NN EEEEEEEEEE TTTTTTTTTT UU UU SSSSSSSS RRRRRRRR  
 NN NN EEEEEEEEEE TTTTTTTTTT UU UU SSSSSSSS RRRRRRRR  
 NN NN EE TT UU UU SS RR RR  
 NN NN EE TT UU UU SS RR RR  
 NNNN NN EE TT UU UU SS RR RR  
 NNNN NN EE TT UU UU SS RR RR  
 NN NN NN EEEEEEEE TT UU UU SSSSSS RRRRRRRR  
 NN NN NN EEEEEEEE TT UU UU SSSSSS RRRRRRRR  
 NN NNNN EE TT UU UU SS RR RR  
 NN NNNN EE TT UU UU SS RR RR  
 NN NN EE TT UU UU SS RR RR  
 NN NN EE TT UU UU SS RR RR  
 NN NN EEEEEEEEEE TT UUUUUUUUUU SSSSSSSS RR RR  
 NN NN EEEEEEEEEE TT UUUUUUUUUU SSSSSSSS RR RR

|          |          |          |
|----------|----------|----------|
| SSSSSSSS | DDDDDDDD | LL       |
| SSSSSSSS | DDDDDDDD | LL       |
| SS       | DD       | DD       |
| SSSSSS   | DD       | DD       |
| SSSSSS   | DD       | DD       |
| SS       | DD       | DD       |
| SSSSSSSS | DDDDDDDD | LLLLLLLL |
| SSSSSSSS | DDDDDDDD | LLLLLLLL |

```
{  
{ NETUSR.SDL - system definitions for NETWORK ACP interface  
{ 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 System Macro Libraries  
{ ABSTRACT:  
{  
 This file contains the SDL source for NETWORK control blocks.  
{ ENVIRONMENT:  
{  
 n/a
```

```
{ AUTHOR: The VMS Group CREATION DATE: 1-Aug-1976
```

```
{ MODIFICATION HISTORY:
```

|      |         |                                                         |             |
|------|---------|---------------------------------------------------------|-------------|
| V024 | RNG0024 | Rod Gamache                                             | 28-Feb-1984 |
|      |         | Add IPID field for LLI database.                        |             |
| V023 | MKP0001 | Kathy Perko                                             | 06-Dec-1983 |
|      |         | Add Node parameter, SERVICE NODE VERSION, for down line |             |
|      |         | load.                                                   |             |
| V022 | TMH0022 | Tim Halvorsen                                           | 04-Jul-1983 |
|      |         | Add XAI database for X.25 gateway support.              |             |
|      |         | Add LNI ALIAS (cluster node name) parameter.            |             |
| V021 | TMH0021 | Tim Halvorsen                                           | 20-Apr-1983 |

Add Service (DLE) database.

V020 RNG0020 Rod Gamache 18-Apr-1983  
Reserve some more 'database' codes for PSI.

V019 TMH0019 Tim Halvorsen 04-Mar-1983  
Add DEVNAM circuit parameter (action routine).

V018 TMH0018 Tim Halvorsen 14-Feb-1983  
Add line buffer size parameter.

V017 TMH0017 Tim Halvorsen 07-Dec-1982  
Add 'Next node to destination' node name, as well as  
the node number which is already a parameter.  
Add Ethernet protocol type parameter, so that Phase IV  
can be used with another protocol type on the NI.

V016 TMH0016 Tim Halvorsen 05-Nov-1982  
Add area database.

V015 TMH0015 Tim Halvorsen 23-Sep-1982  
Add adjacency database.

V014 TMH0014 Tim Halvorsen 30-Jun-1982  
Update Phase IV line, circuit and node parameters.  
Add line action routine to get the physical device  
name being used by a line (including unit number).  
Add LOOP LINE parameters and function code.  
Change DEST SAD from a string to a longword.  
Change DEST NODE from a string to a longword.

V013 TMH0013 Tim Halvorsen 16-Jun-1982  
Add SPI (Server Process) database and parameters.  
Add DECLSERV function code.  
Make NFB\$C\_LENGTH equal to NFB\$L\_FLDID, and not be  
4 bytes afterwards, since it should be possible to  
issue a ACPCONTROL function with NO fields, and without  
any ENDOFLIST field (since it isn't required anymore).

V012 TMH0012 Tim Halvorsen 04-Apr-1982  
Remove STRT KEY. Remove ERR\_STRT detail code.  
Remove NFB\$C\_COLLATE generic field ID.  
Increase size of P2 context area to 64 bytes.  
Remove obsolete function codes.  
Rename most of the symbols and fields to make the interface  
more understandable:  
NFB\$V\_KNO --> NFB\$V\_MULT  
NFB\$V\_UPD --> NFB\$V\_ERRUPD  
NFB\$V\_NOUPD --> NFB\$V\_NOCTX  
NFB\$C\_P2STRTLNG --> NFB\$C\_CTX\_SIZE  
Move \$NDBDEF to NETCTL.MDL  
Add CNT (counters) parameter for XDI, XS5 and XS9 databases.  
Remove fields which were commented out.  
Fix classification of CRI/PLI parameters in comment field.  
Add key and oper fields for second search key.  
Remove obsolete NUL parameters from all databases.

Add Ethernet parameters.  
Remove obsolete FNDNEXT operator, and make FNDMIN and FNDMAX unavailable through the QIO interface, since they don't work.  
Add ERR\_P5 and ERR\_P6 error detail codes, so that they can be returned if P5 or P6 were specified.  
Add circuit DLM flag, to indicate X.25 DLM access (corresponds to the NICE OWNER parameter).  
Add ERR\_FLAGS and ERR\_OPER2 error detail codes.  
Add symbols to make accessing field IDs and values easier for BLISS programs.  
Add DTE MAXIMUM CIRCUITS parameter.

- |        |         |                                                                 |             |
|--------|---------|-----------------------------------------------------------------|-------------|
| V3-011 | TMH0011 | Tim Halvorsen                                                   | 25-Feb-1982 |
|        |         | Add X.25 parameters.                                            |             |
| V3-010 | ADE0033 | A.Eldridge                                                      | 01-Mar-1982 |
|        |         | Increased NFB\$C_DB_MAX to allow access to the X25 databases.   |             |
| V3-009 | ADE0032 | A.Eldridge                                                      | 15-Feb-1982 |
|        |         | Added 'pipeline quota' (PIQ) to the LNI database                |             |
| V3-008 | ADE0031 | A.Eldridge                                                      | 06-Jan-1982 |
|        |         | Removed the 'retransmit timer' (RTT) from the circuit database. |             |
| V3-007 | ADE0030 | Al Eldridge                                                     | 30-Nov-1981 |
|        |         | Added support for proxy logins                                  |             |
| V3-006 | ??      |                                                                 |             |
| V3-005 | ADE0005 | Al Eldridge                                                     | 2-Sep-1981  |
|        |         | Update NFBDEF to get rid of overlapping symbols                 |             |
| V3-004 | TMH0004 | Tim Halvorsen                                                   | 30-Aug-1981 |
|        |         | Get rid of obsolete definitions and add \$NFBDEF.               |             |

---

```

{
{ Network Function Block
{ This is the format of the P1 buffer used for NETACP's IOS$_ACPCONTROL Qio's.
}

module $NFBDEF;

constant ENDOFLIST      equals 0  prefix NFB tag $C;
constant WILDCARD        equals 1  prefix NFB tag $C;
constant CTX_SIZE         equals 64 prefix NFB tag $C; /* Length of context area in P2 buffer

/*
/* The following generic field identifiers are defined for all database
/* Used to terminate the field i.d.
/* Field i.d. used for "match all" database searches
/* Length of context area in P2 buffer

/*
/* The following codes are passed in the second IOSB longword to qual
/* as SSS$_ILLCNTRFUNC error.
/*
/* The high order word of these error codes must be 0
/* so that they won't be confused with field i.d.s

constant(
    FCT
, DB
, P1
, P2
, P3
, P4
, P5
, P6
, CELL
, OPER
, SRCH
, SRCH2
, OPER2
, FLAGS
) equals 1 increment 1 prefix NFB$_E tag RR;
                           /* Unrecognized NFB$B_FCT value.
                           /* Unrecognized NFB$B_DATABASE value.
                           /* The P1 buffer is invalid.
                           /* The P2 buffer is invalid.
                           /* The P3 buffer is invalid.
                           /* The P4 buffer is invalid.
                           /* The P5 buffer should not have been specified.
                           /* The P6 buffer should not have been specified.
                           /* Unrecognized NFB$B_CELL value.
                           /* Unrecognized NFB$B_OPER value.
                           /* Unrecognized NFB$L_SRCH_KEY field ID
                           /* Unrecognized NFB$L_SRCH2_KEY field ID
                           /* Unrecognized NFB$B_OPER2 value.
                           /* Undefined bits in NFB$B_FLAGS were not zero.

/*
/* Define the P1 buffer format
/*

aggregate NFBDEF structure fill prefix NFB$;
    FCT byte unsigned;

constant(
    DECLNAME
, DECLOBJ
, DECLSERV
) equals 21 increment 1 prefix NFB tag $C;
                           /* A function code as follows:
                           /* Function codes for the NFB
                           /* (leaving room for 20 obsolete function codes)
                           /* Declare name
                           /* Declare object
                           /* Declare server process available

                           /* Resume defining function codes
                           /* (leave room for 4 obsolete function codes)
                           /* Log a network event
                           /* Read current raw event queue (used by EVL only)

constant(
    LOGEVENT
, READEVENT
)

```

```

) equals 28 increment 1 prefix NFB tag $C;

constant(
    FC_DELETE           /* Remove an entry from the data base.
    . FC_SHOW            /* Return specified field values.
    . FC_SET             /* Set/modify the field values.
    . FC_CLEAR           /* Clear specified field values.
    . FC_ZERCOU          /* Zero (and optionally read) counters
    . FC_LOOP            /* Loop (used only to PSI to loop an X.25 line)
) equals 33 increment 1 prefix NFB tag $C;      /* Maximum FCT value

constant FC_MAX      equals NFB$C_FC_LOOP  prefix NFB tag $C; /* Maximum FCT value

FLAGS_OVERLAY union fill;
FLAGS byte unsigned;
FLAGS_BITS structure fill;
ERRUPD bitfield mask;
MULT bitfield mask;
NOCTX bitfield mask;

end FLAGS_BITS;

end FLAGS_OVERLAY;
DATABASE byte unsigned;

constant(
    LNI                  /* Local node
    . NDI                /* Common nodes
    . OBI                /* Network objects
    . CRI                /* Circuits
    . PLI                /* Lines
    . EFI                /* Event logging filters
    . ESI                /* Event logging sinks
    . LLI                /* Logical-links
    . XNI                /* X.25 networks
    . XGI                /* X.25 groups
    . XDI                /* X.25 DTEs
    . XS5                /* X.25 server
    . XD5                /* X.25 destinations
    . XS9                /* X.29 server
    . XD9                /* X.29 destinations
    . XTI                /* X.25 trace facility
    . XTT                /* X.25 tracepoints
    . SPI                /* Server Process
    . AJI                /* Adjacency information
    . ARI                /* Area information
    . PSI1               /* (The following codes are reserved for future PSACP
                           /* databases. These codes should only be used in the
                           /* event PSIACP needs a database code before a new
                           /* new NETACP can be supplied to support it).
    . PSI2               /* PSI reserved database
    . PSI3               /* PSI reserved database
    . PSI4               /* PSI reserved database
)

```

```

. PSI4          /* PSI reserved database
. PSI5          /* PSI reserved database
. SDI           /* Service (DLE) information
. XAI           /* X.25 access database
. XXX           /* Last database definition for NFB$C_DB_MAX calc.
} equals 1 increment 1 prefix NFB$C_ tag DB; /* Maximum DATABASE value

constant MAX      equals NFB$C_DB_XXX-1 prefix NFB$C_ tag DB; /* Maximum DATABASE value
OPER byte unsigned;                                     /* Specifies the sense of the search (e.g. EQL, GEQU)
                                                       /* when comparing against the SRCH_KEY field.

constant(
  EQL           /* Match if SEARCH_KEY value EQL database entry field
  . GTRU          /* Match if SEARCH_KEY value GTRU database entry field
  . LSSU          /* Match if SEARCH_KEY value LSSU database entry field
  . NEQ           /* Match if SEARCH_KEY value NEQ database entry field
  . FNDMIN        /* The following may only be used internally by NETACP
  . FNDMAX        /* Find entry with minimum key value
  . FNDPOS        /* Find entry with maximum key value
                  /* Find entry position in database

) equals 0 increment 1 prefix NFB$C_ tag OP; /* Maximum operator function

constant MAXFCT    equals NFB$C_OP_NEQ prefix NFB$C_ tag OP; /* Maximum operator function
constant MAXINT     equals NFB$C_OP_FNDPOS prefix NFB$C_ tag OP; /* Maximum internal function

SRCH_KEY longword unsigned;
                           /* Search key field identifier specifying the key used
                           /* to locate the entry in the database. This search is
                           /* controlled by the sense of the NFB$B_OPER field.
                           /*
                           /* If this field has the value 'NFB$C_WILDCARD', then
                           /* the very next entry in the list is assumed to be the
                           /* target of the search.
                           /*
                           /* If this field is not specified (zero), then it
                           /* is assumed to be NFB$C_WILDCARD (no search key).
                           /*
SRCH2_KEY longword unsigned;
                           /* Secondary search key field ID specifying the key used
                           /* to locate the entry in the database. This search is
                           /* controlled by the sense of the NFB$B_OPER2 field.
                           /*
                           /* If both SRCH_KEY and SRCH2_KEY are specified, then
                           /* only those database entries matching both search keys
                           /* will be processed.
                           /*
                           /* If this field is not specified (zero), then it
                           /* is assumed to be NFB$C_WILDCARD (no search key).
                           /*
OPER2 byte unsigned;
                           /* Specifies the sense of the search (e.g. EQL, GEQU)
                           /* when comparing against the SRCH2_KEY field.
                           /*
MBZ1 byte unsigned;
CELL_SIZE word unsigned;
constant "LENGTH" equals . prefix NFB$ tag K;
constant "LENGTH" equals . prefix NFB$ tag C;
                           /* Reserved. MBZ:
                           /* Some of the field values found in the P4 buffer are
                           /* Minimum structure size.
                           /* Minimum structure size.
                           /* counted strings. If the "cell size" is non-zero, it
                           /* indicates the number of bytes which each string in
                           /* the P4 buffer occupies. If it is zero then strings

```

```
FLDID longword unsigned;
/* fields are stored as variable lengthed strings.
/* Cell containing the first field ID -- the list
/* of field IDs begins here and continues to the
/* end of the structure.
/*
/* The list may be terminated before the end of the
/* structure by placing the value NFB$C_ENDOFLIST
/* in the longword following the last field ID.
/*
/*
/* Define the "field i.d." format.
/*
end NFBDEF;

aggregate NFBDEF1 union fill prefix NFB$;
PARAM ID longword unsigned; /* Define parameter ID longword
PARAM ID BITS structure fill;
INX bitfield mask length 16; /* Index into semantic table
TYP bitfield mask length 2; /* Field type (string, bit, etc.)
SPARE bitfield mask length 6; /* Reserved, MBZ
DB bitfield mask length 8; /* Data-base i.d.
end PARAM ID BITS;
constant TYP_BIT equals 0 prefix NFB tag $C; /* Field type for bits
constant TYP_V equals 0 prefix NFB tag $C; /* Field type for bits
constant TYP_LNG equals 1 prefix NFB tag $C; /* Field type for longwords
constant TYP_L equals 1 prefix NFB tag $C; /* Field type for longwords
constant TYP_STR equals ? prefix NFB tag $C; /* Field type for strings
constant TYP_S equals 2 prefix NFB tag $C; /* Field type for strings

/*
/* Define useful symbols for storing and retrieving binary and string
/* values from the P2 and P4 buffers
/*
end NFBDEF1;

aggregate NFBDEF2 union fill prefix NFB$;
LNG VALUE longword unsigned; /* Longword value
end NFBDEF2;

aggregate NFBDEF3 union fill prefix NFB$;
BIT VALUE longword unsigned; /* Boolean value
end NFBDEF3;

aggregate NFBDEF4 union fill prefix NFB$;
STR_COUNT word unsigned; /* String count field
STR_COUNT_FIELDS structure fill;
FILL_T byte dimension 2 fill prefix NFBDEF tag $$;
STR_TEXT character length 0 tag B; /* Start of string data

/*
/* Define identifiers for each parameter in all database
/*
/* ** The low order 16 bits for each parameter must be unique **
/* *** with respect to all other parameters in its particular ***

```

NETUSR.SDL;1

16-SEP-1984 16:42:34.33 <sup>J</sup> 1 Page 8

/\* \*\* database.

\*\*

```
/*
/* Define a field identifier index for each parameter in the NDI database.
*/
/*
/* Boolean parameters
*/
constant(
    LCK
, LOO
, REA
) equals (((NFB$C_DB_NDI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_N tag DI;

/*
/* "Longword" Parameters
*/
constant(
    TAD
, CTA
, ADD
, CTI
, ACL
, DEL
, DTY
, DCO
, DHO
, SDV
, CPU
, STY
, DAD
, DCT
, OHO
, IHO
, ACC
, PRX
, NND
, SNV
) equals (((NFB$C_DB_NDI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_N tag DI;

/*
/* String parameters
*/
constant(
    COL
, HAC
, CNT
, NNA
, SLI
, SPA
, LOA
, SLO
, TLO
, SID
, DUM
) equals (((NFB$C_DB_NDI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_N tag DI;
```

```
. SDU          /* Secondary dumper
. NLI          /* Loopback Line
. DLI          /* Destination Line
. PUS          /* Privileged user id
. PAC          /* Privileged account
. PPW          /* Privileged password
. NUS          /* Non-privileged user id
. NAC          /* Non-privileged account
. NPW          /* Non-privileged password
. RPA          /* Receive password
. TPA          /* Transmit password
. DFL          /* Diagnostic load file
. HWA          /* Hardware NI address (ROM address)
. LPA          /* Loop assistant NI address
. NNN          /* Next node name to destination (goes with NND)
} equals (((NFB$C_DB_NDI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_N tag DI;
```

```
/*
 * Define a field identifier index for each parameter in the LNI database.
 */
/*
 * Boolean parameters
 */
constant(
    LCK           /* Set if conditionally writable fields are not writable
    SUP           /* Set if area numbers are to be suppressed
) equals (((NFB$C_DB_LNI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_L tag NI;

/*
 * "Longword parameters
 */
constant(
    ADD           /* Address
    . ACL          /* Total number of active links
    . ITI          /* Incoming timer
    . OTI          /* Outgoing timer
    . STA          /* State
    . MLK          /* Maximum links
    . DFA          /* Delay factor
    . DWE          /* Delay weight
    . IAT          /* Inactivity timer
    . RFA          /* Retransmit factor
    . ETY          /* Executor Type
    . RTI          /* Routing timer
    . RSI          /* Routing suppression timer
    . SAD          /* Subaddress
                    /* (lower word = lower limit, upper word = upper limit)
    . MAD          /* Maximum address
    . MLN          /* Maximum lines
    . MCO          /* Maximum cost
    . MHO          /* Maximum hops
    . MVI          /* Maximum visits
    . MBU          /* Maximum buffers
    . BUS          /* Forwarding buffer size
    . LPC          /* Loop count
    . LPL          /* Loop length
    . LPD          /* Loop Data type
    . DAC          /* Default access switch (inbound, outbound, etc)
    . DPX          /* Default proxy access (inbound, outbound, etc)
    . PIQ          /* Pipeline quota
    . LPH          /* Loop help type of assistance given to loop requestors
    . BRT          /* Broadcast routing timer
    . MAR          /* Maximum areas
    . MBE          /* Maximum nonrouters on NI
    . MBR          /* Maximum routers on NI
    . AMC          /* Area maximum cost
    . AMH          /* Area maximum hops
    . SBS          /* Segment buffer size
    . ALI          /* Alias local node address (cluster address)
) equals (((NFB$C_DB_LNI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_L tag NI;
/*
```

```
    /* String parameters
     */
constant(
    COL          /* Collating field
    . NAM         /* Local node name
    : CNT         /* Counters
    : IDE         /* Identification
    , MVE         /* Management version
    , NVE         /* Nsp version
    , RVE         /* Routing version
    PHA          /* Physical NI address (current address)
) equals (((NFB$C_DB_LNI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_L tag NI;
```

```
/*
 * Define a field identifier index for each parameter in the OBI database.
 */
/*
 */
/* Boolean Parameters
*/
constant(
    LCK
    . SET
) equals (((NFB$C_DB_OBI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_0 tag BI;

/*
 */
/* Longword Parameters
*/
constant(
    LPR
    . HPR
    . UCB
    . CHN
    . NUM
    . PID
    . PRX
) equals (((NFB$C_DB_OBI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_0 tag BI;

/*
 */
/* String Parameters
*/
constant(
    COL
    . ZNA
    . SFI
    . IAC
    . NAM
    . FID
    . USR
    . ACC
    . PSW
) equals (((NFB$C_DB_OBI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_0 tag BI;
```



```
    . IAB          /* D Inactive base
    . IAI          /* D Inactive increment
    . IAT          /* D Inactive threshold
    . DYB          /* D Dying base
    . DYI          /* D Dying increment
    . DYT          /* D Dying threshold
    . DTH          /* D Dead threshold
    . MST          /* D Maintenance mode state (0 => On, 1 => Off)
    . XPT          /* E Transport protocol to use
    . MRT          /* E Maximum routers on this NI
    . RPR          /* E Router priority
    . DRT          /* E Designated router on NI (node address)
} equals (((NFB$C_DB_CRI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_C tag RI;

/*
/* String Parameters
*/
constant(
    COL          /* D Collating field
    . NAM          /* C Circuit name
    . VMSNAM       /* D Device name in VMS format
    . CHR          /* D Characteristics buffer for startup control QIO
    . CNT          /* C Counters
    . P2P          /* D Line's PhaseII partner name (for loopback)
    . LOO          /* E Loopback name
    . PNN          /* E Adjacent node name
    . NUM          /* X Call Number
    . DTE          /* X DTE
    . DEVNAM       /* D Device name in VMS format, with unit included
} equals (((NFB$C_DB_CRI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_C tag RI;
```

```
/*
 * Define a field identifier index for each parameter in the PLI database.
 */
/* C = common
 * L = LAPB (X.25)
 * D = DDNCMP (not X.25)
 * E = Ethernet
 */
/* Use
 */
/* ----
 */
/* Boolean Parameters
 */

constant(
    LCK
    . SER
    . DUP
    . CON
    . CLO
) equals (((NFB$C_DB_PLI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_P tag LI;

/*
 * "Longword" Parameters
 */

constant(
    CTA
    . STA
    . SUB
    . LCT
    . PRO
    . STI
    . HTI
    . MBL
    . MRT
    . MWI
    . SLT
    . DDT
    . DLT
    . SRT
    . BFN
    . BUS
    . PLVEC
    . RTT
    . MOD
    . LPC
    . LPL
    . LPD
    . EPT
    . BFS
) equals (((NFB$C_DB_PLI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_P tag LI;

/*
 * String Parameters
 */
```

```
constant(
    COL                      /* D Collating field
    . NAM                     /* C Line name
    . VMSNAM                  /* D Device name in VMS format
    . CHR                     /* D Set-mode $QIO Line Characteristics buffer
    . CNT                     /* C Counters
    . MCD                     /* L Filespec for microcode dump (initiates dump)
    . HWA                     /* D NI hardware address (ROM address)
    . DEVNAM                  /* D Device name in VMS format, with unit included
) equals (((NFB$C_DB_PLI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_P tag LI;
```

```
/*
 * Define a field identifier index for each parameter in the EFI database.
 */
/*
 */
/* Boolean Parameters
 */
constant(
    LCK
) equals (((NFB$C_DB_EFI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_E tag FI;

/*
 */
/* "Longword" Parameters
 */
constant(
    SIN
, SP1
, B1
, B2
) equals (((NFB$C_DB_EFI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_E tag FI;

/*
 */
/* String Parameters
 */
constant(
    COL
, EVE
, SB1
, SB2
, SB3
) equals (((NFB$C_DB_EFI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_E tag FI;
```

```
/*
 * Define a field identifier index for each parameter in the ESI database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK
) equals (((NFB$C_DB_ESI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_E tag SI;

/*
 * "Longword" Parameters
 */
constant(
    SNK
, STA
, SP1
, B1
, B2
) equals (((NFB$C_DB_ESI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_E tag SI;

/*
 * String Parameters
 */
constant(
    COL
, LNA
, SB1
, SB2
, SB3
) equals (((NFB$C_DB_ESI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_E tag SI;
```

```
/*
 * Define a field identifier index for each parameter in the LLI database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK
) equals (((NFB$C_DB_LLI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_L tag LI;

/*
 * Longword Parameters
 */
constant(
    DLY
    STA
    LLN
    RLN
    PNA
    PID
    IPID
    XWB
    CNT
) equals (((NFB$C_DB_LLI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_L tag LI;

/*
 * String Parameters
 */
constant(
    COL
    USR
    PRC
    PNN
    RID
) equals (((NFB$C_DB_LLI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_L tag LI;
```

```
/*
 * X.25 network parameters (part of MODULE X25-PROTOCOL)
 */
/* Define a field identifier index for each parameter in the XNI database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK
    MNS
) equals (((NFB$C_DB_XNI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag NI;

/*
 * "Longword" Parameters
 */
constant(
    CAT
    CLT
    DBL
    DWI
    MBL
    MCL
    MRS
    MST
    MWI
    RST
    STT
) equals (((NFB$C_DB_XNI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag NI;

/*
 * String Parameters
 */
constant(
    COL
    NET
) equals (((NFB$C_DB_XNI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag NI;
```

```
/*
 * X.25 DTE parameters (qualified by a given network)
 */
/* Define a field identifier index for each parameter in the XDI database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK
) equals (((NFB$C_DB_XDI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag DI;

/*
 * "Longword" Parameters
 */
constant(
    ACH
, ASW
, CTM
, MCH
, STA
, SUB
, MCI
) equals (((NFB$C_DB_XDI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag DI;

/*
 * String Parameters
 */
constant(
    COL
, DTE
, CHN
, LIN
, NET
, CNT
) equals (((NFB$C_DB_XDI@24)+(NFB$C_TYP_STRA16)+64)) increment 1 prefix NFB$C_X tag DI;
```

```
/* X.25 group parameters (qualified by a given DTE)
/* Define a field identifier index for each parameter in the XGI database.
/*
/* Boolean Parameters
/*
constant(
    LCK
) equals (((NFB$C_DB_XGI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag GI;

/*
/* "Longword" Parameters
/*
constant(
    GNM
    GTY
) equals (((NFB$C_DB_XGI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag GI;

/*
/* String Parameters
/*
constant(
    COL
    . GRP
    GDT
) equals (((NFB$C_DB_XGI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag GI;

/* Collating field. This field must be unique across
/* all entries in this database. It consists of the
/* group-name string followed by the DTE address.
/* Group name
/* Group DTE address
```

```
/*
 * X.25 server parameters (global parameters for all destinations)
 */
/* Define a field identifier index for each parameter in the XS5 database.
 */
/*
 * Boolean Parameters
 */

constant(
    LCK
) equals (((NFB$C_DB_XS5@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag S5;

/*
 * "Longword" Parameters
 */

constant(
    MCI
, STA
, ACI
, CTM
) equals (((NFB$C_DB_XS5@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag S5;

/*
 * String Parameters
 */

constant(
    COL
, CNT
) equals (((NFB$C_DB_XS5@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag S5;

/*
 * Collating field. This field must be unique across
 * all entries in this database.
 */
/*
 * Counters
 */
```

```
/*
 * X.25 destination parameters (part of MODULE X25-SERVER)
 */
/* Define a field identifier index for each parameter in the XDS database.
*/
/*
 * Boolean Parameters
*/
constant(
    LCK
) equals (((NFB$C_DB_XD5@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag D5;

/*
 * "Longword" Parameters
*/
constant(
    PRI
, SAD
, NOD
) equals (((NFB$C_DB_XD5@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag D5;

/*
 * String Parameters
*/
constant(
    COL
, DST
, CMK
, CVL
, GRP
, NUM
, OBJ
, FIL
, USR
, PSW
, ACC
) equals (((NFB$C_DB_XD5@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag D5;

/*
 * Collating field. This field must be unique across
 * all entries in this database.
*/
/*
 * Destination DTE address
*/
/*
 * Call mask
*/
/*
 * Call value
*/
/*
 * Group name
*/
/*
 * DTE number
*/
/*
 * && Object name
*/
/*
 * Command procedure to execute when starting object
*/
/*
 * User name
*/
/*
 * Password
*/
/*
 * Account
*/
```

```
/*
 * X.29 server parameters (global parameters for all destinations)
 */
/* Define a field identifier index for each parameter in the XS9 database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK
) equals (((NFB$C_DB_XS9@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag S9;

/*
 * "Longword" Parameters
 */
constant(
    MCI
, STA
, ACI
, CTM
) equals (((NFB$C_DB_XS9@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag S9;

/*
 * String Parameters
 */
constant(
    COL
, CNT
) equals (((NFB$C_DB_XS9@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag S9;
```

```
/*
 * X.29 destination parameters (part of MODULE X29-SERVER)
 */
/* Define a field identifier index for each parameter in the XD9 database.
*/
/*
 * Boolean Parameters
*/
constant(
    LCK /* Set if conditionally writable fields are not writable
) equals (((NFB$C_DB_XD9@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag D9;

/*
 * "Longword" Parameters
*/
constant(
    PRI /* Priority
    . SAD /* Subaddress range
    . NOD /* (lower word = lower limit, upper word = upper limit)
            /* Remote node address containing server (gateways only)
) equals (((NFB$C_DB_XD9@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag D9;

/*
 * String Parameters
*/
constant(
    COL /* Collating field. This field must be unique across
        /* all entries in this database.
    , DST /* Destination DTE address
    , CMK /* Call mask
    , CVL /* Call value
    , GRP /* Group name
    , NUM /* DTE number
    , OBJ /* && Object name
    , FIL /* Command procedure to execute when starting object
    , USR /* User name
    , PSW /* Password
    , ACC /* Account
) equals (((NFB$C_DB_XD9@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag D9;
```

```
/*
/* X.25 tracing facility (global) parameters.
/*
/* Define a field identifier index for each parameter in the XTI database.
/*
/*
/* Boolean Parameters
/*
constant(
    LCK /* Set if conditionally writable fields are not writable
) equals (((NFB$C_DB_XTI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag TI;

/*
/* "Longword" Parameters
/*
constant(
    STA /* State
    . BFZ /* Buffer size
    ; CPL /* Capture limit
    ; MBK /* Maximum blocks/file
    ; MBF /* Maximum number of buffers
    ; MVR /* Maximum trace file version number
) equals (((NFB$C_DB_XTI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag TI;

/*
/* String Parameters
/*
constant(
    COL /* Collating field. This field must be unique across
    ; FNM /* all entries in this database.
    ; Trace file name
) equals (((NFB$C_DB_XTI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag TI;
```

```
/*
 * X.25 tracpoint (local) parameters.
 */
/* Define a field identifier index for each parameter in the XTT database.
 */
/*
 * Boolean Parameters
 */
constant(
    LCK /* Set if conditionally writable fields are not writable
) equals (((NFB$C_DB_XTT@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag TT;

/*
 * "Longword" Parameters
 */
constant(
    TST /* State
    CPS /* Capture size
) equals (((NFB$C_DB_XTT@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag TT;

/*
 * String Parameters
 */
constant(
    COL /* Collating field. This field must be unique across
        /* all entries in this database.
    TPT /* Tracepoint name
) equals (((NFB$C_DB_XTT@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag TT;
```

```
/*
 * X.25 Access (qualified by a given network)
 */
/* Define a field identifier index for each parameter in the XAI database.
*/
/*
 * Boolean Parameters
*/
constant(
    LCK /* Set if conditionally writable fields are not writable
) equals (((NFB$C_DB_XAI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_X tag AI;

/*
 * "Longword" Parameters
*/
constant(
    NDA /* Node address
) equals (((NFB$C_DB_XAI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_X tag AI;

/*
 * String Parameters
*/
constant(
    COL /* Collating field
    . NET /* Network
    . USR /* User id
    . PSW /* Password
    . ACC /* Account
    NOD /* Node id
) equals (((NFB$C_DB_XAI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_X tag AI;
```

```
/*
/* Define SPI (Server Process) parameters
*/
/*
/* Boolean Parameters
*/
constant(
    LCK
    PRL
) equals (((NFB$C_DB_SPI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_S tag PI;

/*
/* Longword Parameters
*/
constant(
    PID
    , IRP
    ; CHN
    ; RNA
) equals (((NFB$C_DB_SPI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_S tag PI;

/*
/* String Parameters
*/
constant(
    COL
    , ACS
    ; RID
    ; SFI
    ; NCB
    ; PNM
) equals (((NFB$C_DB_SPI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_S tag PI;
```

```
/*
/* Define AJI (Adjacency) parameters
/*
/*
/* Boolean Parameters
/*
constant(
    LCK
    REA
) equals (((NFB$C_DB_AJI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_A tag JI;

/*
/* Longword Parameters
/*
constant(
    ADD
    . TYP
    . LIT
    . BLO
    . RPR
) equals (((NFB$C_DB_AJI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_A tag JI;

/*
/* String Parameters
/*
constant(
    COL
    . NNA
    . CIR
) equals (((NFB$C_DB_AJI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_A tag JI;
```

```
/*
 * Define SDI (Service DLE) parameters
 */
/*
 * Boolean Parameters
 */

constant(
    LCK /* Set if conditionally writable fields are not writable
) equals (((NFB$C_DB_SDI@24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_S tag DI;

/*
 * Longword Parameters
 */

constant(
    SUB /* Service substate
    PID /* PID of process owning this DLE link
) equals (((NFB$C_DB_SDI@24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_S tag DI;

/*
 * String Parameters
 */

constant(
    COL /* Collating field
    CIR /* Circuit name
    PHA /* Service physical address (BC only)
    PRC /* Name of process owning this DLE link
) equals (((NFB$C_DB_SDI@24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_S tag DI;
```

```
/*
 * Define the AREA database (read only) for level 2 Phase IV routers only.
 */
/*
 */
/*
 * Boolean parameters
*/
constant(
    LCK           /* Set if conditionally writable fields are not writable
    REA           /* Set if node is reachable
) equals (((NFB$C_DB_ARIA24)+(NFB$C_TYP_BIT@16)+1)) increment 1 prefix NFB$C_A tag RI;

/*
 */
/*
 * "Longword" Parameters
*/
constant(
    ADD           /* Address
    DCO           /* Destination Cost
    DHO           /* Destination Hops
    NND           /* Next node address
) equals (((NFB$C_DB_ARIA24)+(NFB$C_TYP_LNG@16)+16)) increment 1 prefix NFB$C_A tag RI;

/*
 */
/*
 * String parameters
*/
constant(
    COL           /* Collating field
    DLI           /* Circuit used for normal traffic to area
) equals (((NFB$C_DB_ARIA24)+(NFB$C_TYP_STR@16)+64)) increment 1 prefix NFB$C_A tag RI;

end STR_COUNT_FIELDS;
end NFBDEF4;
end_module $NFBDEF;
module $DRDEF;
```

```
/*
/* DISCONNECT REASONS
*/

constant DR_NORMAL      equals 0  prefix NET tag $C:    /* NO ERROR (SYNCH DISCONNECT)
constant DR_RSU           equals 1  prefix NET tag $C:    /* COULDN'T ALLOCATE UCB ADDRESS
constant DR_NONODE        equals 2  prefix NET tag $C:    /* Unrecognized node name
constant DR_SHUT          equals 3  prefix NET tag $C:    /* NODE OR LINE SHUTTING DOWN
constant DR_NOBJ          equals 4  prefix NET tag $C:    /* UNKNOWN OBJECT TYPE OR PROCESS
constant DR_FMT            equals 5  prefix NET tag $C:    /* ILLEGAL PROCESS NAME FIELD
constant DR_BUSY          equals 6  prefix NET tag $C:    /* Object too busy
constant DR_PROTCL        equals 7  prefix NET tag $C:    /* GENERAL PROTOCOL ERROR
constant DR_THIRD          equals 8  prefix NET tag $C:    /* THIRD PARTY DISCONNECT
constant DR_ABORT          equals 9  prefix NET tag $C:    /* DISCONNECT ABORT
constant DR_IVNODE         equals 21 prefix NET tag $C:   /* Invalid node name format
constant DR_NONZ           equals 21 prefix NET tag $C:   /* NON-ZERO DST ADDRESS
constant DR_BADLNK          equals 22 prefix NET tag $C:   /* INCONSISTENT DSTLNK
constant DR_ZERO            equals 23 prefix NET tag $C:   /* ZERO SOURCE ADDRESS
constant DR_BADFC          equals 24 prefix NET tag $C:   /* FCVAL ILLEGAL
constant DR_NOCON          equals 32 prefix NET tag $C:   /* NO CONNECT SLOTS AVAILABLE
constant DR_ACCESS          equals 34 prefix NET tag $C:   /* INVALID ACCESS CONTROL
constant DR_BADSRV          equals 35 prefix NET tag $C:   /* LOGICAL LINK SERVICES MISMATCH
constant DR_ACCNT          equals 36 prefix NET tag $C:   /* INVALID ACCOUNT INFORMATION
constant DR_SEGSIZ          equals 37 prefix NET tag $C:   /* SEGSIZE TOO SMALL
constant DR_EXIT            equals 38 prefix NET tag $C:   /* USER EXIT OR TIMEOUT
constant DR_NOPATH          equals 39 prefix NET tag $C:   /* NO PATH TO DESTINATION NODE
constant DR_LOSS             equals 40 prefix NET tag $C:   /* LOSS OF DATA HAS OCCURRED
constant DR_NOLINK          equals 41 prefix NET tag $C:   /* ILLEGAL MSG FOR LINK NOLINK STATE
constant DR_CONF             equals 42 prefix NET tag $C:   /* REAL DISCONNECT CONFIRM
constant DR_IMLONG          equals 43 prefix NET tag $C:   /* IMAGE DATA FIELD TOO LONG

end_module $DRDEF;
```

0273 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

NETUSR  
SOL

LIBTAIR  
B32

XWBDEF  
SOL

NETDEFS  
MAR

PSIUSR  
SOL

NETDRVMAC  
MAR

NDRIVER  
LIS

NSPMSGDEF  
SOL

LIBHEAD  
B32

NETMACROS  
MAR

NET  
LIS

NETACPTRN  
LIS