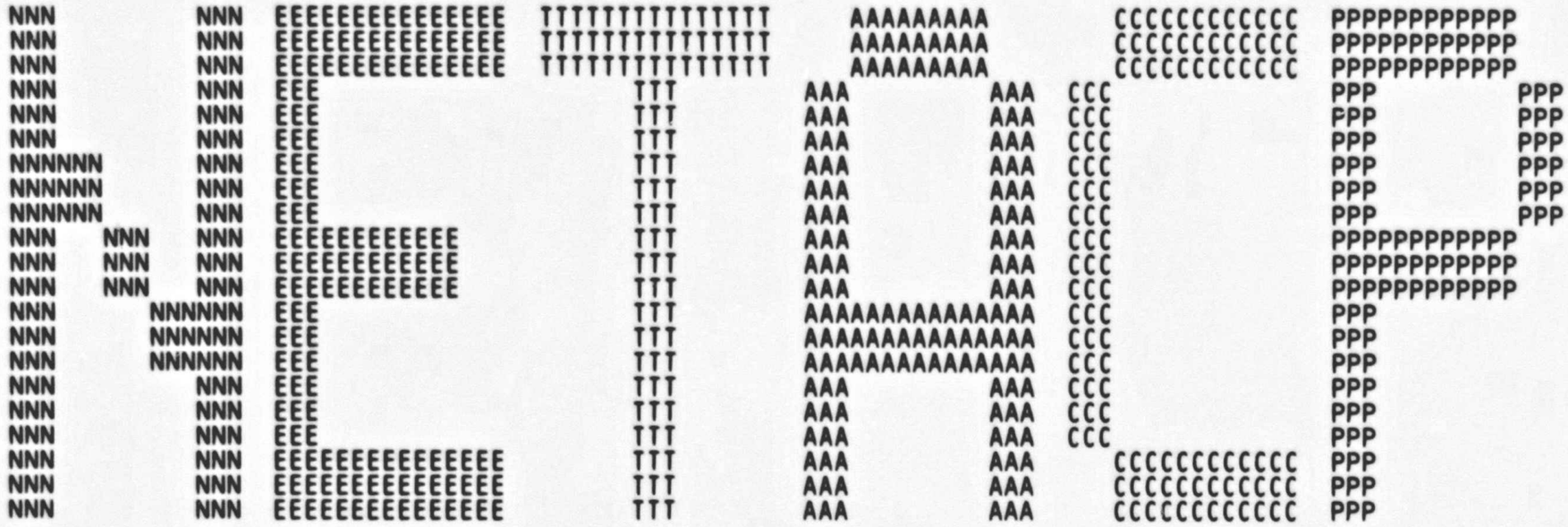
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN 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 PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
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**NETMACROS

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  MM      MM    AAAAAA     CCCCCCCC  RRRRRRRR     OOOOOO     SSSSSSSS
NN      NN  EEEEEEEEEE  TTTTTTTTTT  MM      MM    AAAAAA     CCCCCCCC  RRRRRRRR     OOOOOO     SSSSSSSS
NN      NN  EE              TT      MMMM  MMMM   AA    AA   CC        RR      RR   OO    OO   SS
NN      NN  EE              TT      MMMM  MMMM   AA    AA   CC        RR      RR   OO    OO   SS
NNNN    NN  EE              TT      MM MMMM MM   AA    AA   CC        RR      RR   OO    OO   SS
NNNN    NN  EE              TT      MM MMMM MM   AA    AA   CC        RR      RR   OO    OO   SS
NN  NN  NN  EEEEEEEE        TT      MM  MM  MM   AA    AA   CC        RRRRRRRR     OO    OO    SSSSSS
NN  NN  NN  EEEEEEEE        TT      MM  MM  MM   AA    AA   CC        RRRRRRRR     OO    OO    SSSSSS
NN    NNNN  EE              TT      MM      MM   AAAAAAAAAA CC        RR  RR       OO    OO          SS
NN    NNNN  EE              TT      MM      MM   AAAAAAAAAA CC        RR  RR       OO    OO          SS
NN      NN  EE              TT      MM      MM   AA    AA   CC        RR    RR     OO    OO          SS
NN      NN  EE              TT      MM      MM   AA    AA   CC        RR    RR     OO    OO          SS
NN      NN  EEEEEEEEEE      TT      MM      MM   AA    AA    CCCCCCCC RR     RR    OOOOOO    SSSSSSSS
NN      NN  EEEEEEEEEE      TT      MM      MM   AA    AA    CCCCCCCC RR     RR    OOOOOO    SSSSSSSS
```

```
MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM   AA    AA   RR      RR
MMMM  MMMM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AAAAAAAAAA RR  RR
MM      MM   AAAAAAAAAA RR  RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR      RR
MM      MM   AA    AA   RR      RR
```

```
        .TITLE  NETMACROS - Common NETACP/NETDRIVER macro definitions
        .IDENT  'V04-000'
```



```
; MODIFIED BY:
;
;       V006    RNG0006         Rod Gamache             20-Apr-1983
;               Make $GETFLD, $PUTFLD, $SEARCH and $CLRFLD macros always
;               use JSB's to routines.
;
;       V005    RNG0005         Rod Gamache             30-Mar-1982
;               Add PUSHQ and POPQ macros. Fix the $DISPATCH macro to
;               not attempt to generate CASE statements for symbols
;               greater than 64k. Alter the $SEARCH macro to generate
;               longword JSBs instead of BSBWs in special cases.
;               Alter the $GETFLD, $PUTFLD and $CLRFLD macros to generate
;               longword JSBs instead of BSBWs in special cases.
;
;       V004    TMH0004         Tim Halvorsen           05-May-1982
;               Add SETBIT and CLRBIT macros.
;
;       V02-003 ADE003          A.ELDRIDGE              22-Jan-1982
;               Changed offsets in $DISPATCH to used signed_word.
;-
```

```
;
; Macro to build a CNF field identifier
;
.MACRO  .CNFFLD  cnf_id,type,field
        .LONG   NFB$C_'cnf_id'_'field'
.ENDM   .CNFFLD

.MACRO  $CNFFLD  cnf_id,type,field,dst
        MOVL    #NFB$C_'cnf_id'_'field',dst
.ENDM   $CNFFLD

;
;  Macros to call CNF functions
;
.MACRO  $SEARCH  op,cnf,typ,fld
        ASSUME  NFB$C_OP_EQL EQ 0
        .IF     IDENTICAL op eql
        .IF_TRUE
                CLRL    R1
        .IF_FALSE
                MOVL    S^#NFB$C_OP_'op',R1
        .ENDC
        $CNFFLD  cnf,typ,fld,R9
        JSB     CNF$KEY_SEARCH
.ENDM   $SEARCH

.MACRO  $CLRFLD  cnf,typ,fld
        $CNFFLD  cnf,typ,fld,R9
        JSB     CNF$CLR_FIELD
.ENDM   $CLRFLD

.MACRO  $PUTFLD  cnf,typ,fld
        $CNFFLD  cnf,typ,fld,R9
        JSB     CNF$PUT_FIELD
.ENDM   $PUTFLD
```

```
.MACRO  $GETFLD  cnf,typ,fld

        $CNFFLD  cnf,typ,fld,R9

        JSB     CNF$GET_FIELD

.ENDM   $GETFLD

;
; Macro to update a performance counter
;
.MACRO  BUMP    WIDTH,ADDRESS,?LL
        INC'WIDTH  ADDRESS
        BVC        LL
        MNEG'WIDTH #1,ADDRESS
    LL:
.ENDM   BUMP

.MACRO  UPDATE  WIDTH,INCR,ADDRESS,?LL
        ADD'WIDTH  INCR,ADDRESS
        BVC        LL
        MNEG'WIDTH #1,ADDRESS
    LL:
.ENDM   UPDATE
```

```
;
; This macro translates into the CASEx instruction.  It calculates the
; "base" and "limit" parameters from the <index,displacement> list
; specfied in the 'vector' parameter.  The dispatch table is set up
; such that any unspecified index value within the bounds of the
; transfer vector is associated with a diplacement which transfers
; control to the first location after the CASE statement, i.e., behaves
; as if the index were out of bounds.
;
; Example:
;       $DISPATCH       R0,<-                           ; Message type in R0
;
;                ;index  displacement
;
;                <CI,    NSP$RCV_CI>,-                   ; Process CI message
;                <CC,    NSP$RCV_CC>,-                   ; Process CC message
;                <DI,    NSP$RCV_DI>,-                   ; Process DI message
;                <DC,    NSP$RCV_DI>,-                   ; Process DC message
;       >
;       BRW     NSP$RCV_ILLMSG                          ; Message type unknown
;
.MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
SS:

        .MACRO  $DSP1,$DSP1_1
                .IRP    $DSP1_2,$DSP1_1
                        $DSP2   $DSP1_2
                .ENDR
        .ENDM

        .MACRO  $DSP2,$DSP2_1,$DSP2_2
                .=<$DSP2_1-MN>*2 + S
                .SIGNED_WORD    $DSP2_2-S
        .ENDM


        .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
                $BND2   $BND1_1,$BND1_2
        .ENDM

        .MACRO  $BND2,$BND2_1,$BND2_2
                .IIF    NE,<$BND2_2 & <^XFFFF>> - $BND2_2, .ERROR $BND2_2 ; Value of $BND2_2 is too large
                .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
        .ENDM

        .MACRO  $BND    $BND_1,$BND_2
                .IRP    $BND_3,<$BND_2>
                        $BND1   $BND_1,$BND_3
                .ENDR
        .ENDM

        .=0
ZZ:
        $BND    GT,<VECTOR>
MX:
        $BND    LT,<VECTOR>
```

```
MN:
        .=SS

CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
S:
        .REPT   MX-MN+1
        .WORD   <MX-MN>*2 + 2
        .ENDR

        .=S

        $DSP1   <<VECTOR>>

        .=<MX-MN>*2 + S + 2

.ENDM
```

```
;
; MACRO TO SET OR CLEAR A BIT BY BIT NUMBER
;
;CALL:
;       SETBIT  BITNUM,FLAGWORD
; OR:
;       CLRBIT  BITNUM,FLAGWORD
;
; WHERE:
;       BITNUM IS ANY VALID SOURCE OPERAND SECIFYING THE BIT
;              OFFSET FROM THE FLAG BASE TO SET/CLEAR
;
;       FLAGWORD IS ANY VALID DESTINATION OPERAND
;
        .MACRO  SETBIT  VAL,FLAG
        .NTYPE   $$     VAL
        .IF EQ  <$$_-^X0EF>
        .IF NDF VAL
        BBSS    S^#VAL,FLAG,.+1
        .IFF
        .IF LT <VAL-8>
        BISB    #<1@VAL>,FLAG
        .IFF
        BBSS    #VAL,FLAG,.+1
        .ENDC
        .ENDC
        .IFF
        BBSS    VAL,FLAG,.+1
        .ENDC
        .ENDM   SETBIT
;
        .MACRO  CLRBIT  VAL,FLAG
        .NTYPE   $$     VAL
        .IF EQ  <$$_-^X0EF>
        .IF NDF VAL
        BBCC    S^#VAL,FLAG,.+1
        .IFF
        .IF LT <VAL-8>
        BICB    #<1@VAL>,FLAG
        .IFF
        BBCC    #VAL,FLAG,.+1
        .ENDC
        .ENDC
        .IFF
        BBCC    VAL,FLAG,.+1
        .ENDC
        .ENDM   CLRBIT
```

```
;
; MACRO TO PUSH OR POP A QUADWORD ONTO THE STACK
;
;CALL:
;       PUSHQ   LOC
; OR:
;       POPQ    LOC
;
; WHERE:
;       LOC IS EITHER A DOUBLE REGISTER SET OR A MEMORY LOCATION TO BE
;               SAVED ON THE STACK
;
        .MACRO  PUSHQ   LOC
        MOVQ    LOC,-(SP)               ; Save LOC on stack
        .ENDM   PUSHQ
;
        .MACRO  POPQ    LOC
        MOVQ    (SP)+,LOC               ; Restore LOC
        .ENDM   POPQ
```

```
        .END
```

NETUSR
SDL

LIBTAIL
B32

XWBDEF
SDL

NETDEFS
MAR

PSIUSR
SDL

NETDRVMAC
MAR

NODRIVER
LIS

NSPMSGDEF
SDL

LIBHEAD
B32

NET
LIS

NETMACROS
MAR

NETACPTRN
LIS