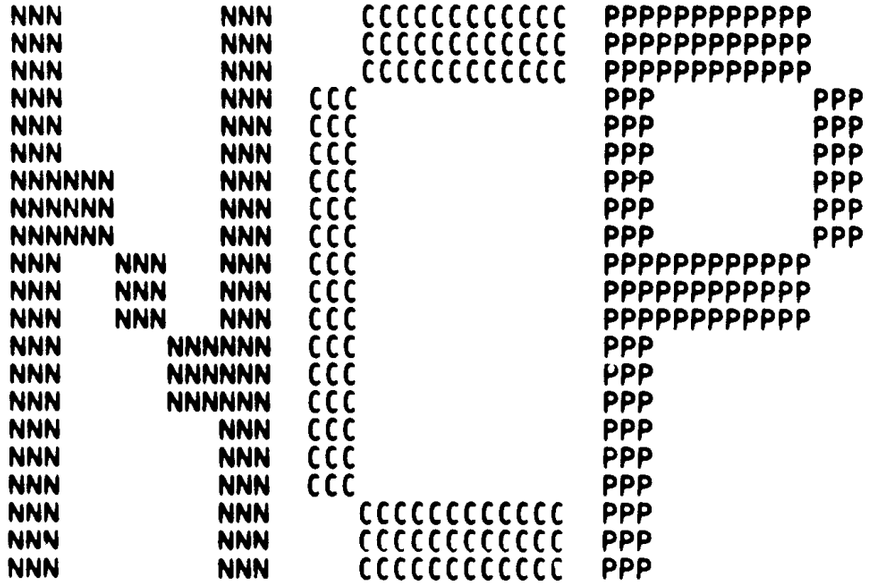
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNNNNN NNN CCC PPP PPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNN NNN CCC PPPPPPPPPPPP
NNN NNNNNN CCC PPP
NNN NNNNNN CCC PPP
NNN NNNNNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCC PPP
NNN NNN CCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCC PPP
NNN NNN CCCCCCCCCCC PPP

```
NN       NN    CCCCCCCC  PPPPPPPP      CCCCCCCC    000000    NN       NN  MM       MM    AAAAAA    NN       NN
NN       NN    CCCCCCCC  PPPPPPPP      CCCCCCCC    000000    NN       NN  MM       MM    AAAAAA    NN       NN
NN       NN  CC          PP      PP  CC          00      00  NN       NN  MMMM   MMMM  AA      AA  NN       NN
NN       NN  CC          PP      PP  CC          00      00  NN       NN  MMMM   MMMM  AA      AA  NN       NN
NNNN     NN  CC          PP      PP  CC          00      00  NNNN     NN  MM  MM  MM  AA      AA  NNNN     NN
NNNN     NN  CC          PP      PP  CC          00      00  NNNN     NN  MM  MM  MM  AA      AA  NNNN     NN
NN  NN   NN  CC          PPPPPPPP    CC          00      00  NN  NN   NN  MM      MM  AA      AA  NN  NN   NN
NN  NN   NN  CC          PPPPPPPP    CC          00      00  NN  NN   NN  MM      MM  AA      AA  NN  NN   NN
NN    NNNN  CC          PP          CC          00      00  NN    NNNN  MM      MM  AAAAAAAAAA  NN    NNNN
NN    NNNN  CC          PP          CC          00      00  NN    NNNN  MM      MM  AAAAAAAAAA  NN    NNNN
NN      NN  CC          PP          CC          00      00  NN      NN  MM      MM  AA      AA  NN      NN      ....
NN      NN  CC          PP          CC          00      00  NN      NN  MM      MM  AA      AA  NN      NN      ....
NN      NN    CCCCCCCC  PP            CCCCCCCC    000000    NN      NN  MM      MM  AA      AA  NN      NN      ....
NN      NN    CCCCCCCC  PP            CCCCCCCC    000000    NN      NN  MM      MM  AA      AA  NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  NCPCONMAN - Console Carrier Requester Network Management
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;++
0000    29 ; FACILITY: REMOTE CONSOLE CARRIER (NI)
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       THIS PROGRAM RUNS ON A LOCAL NODE TO ALLOW A TERMINAL TO APPEAR TO
0000    34 ;       BE LOCALLY CONNECTED TO A REMOTE CONSOLE.
0000    35 ;
0000    36 ;
0000    37 ; ENVIRONMENT: VMS - USER MODE
0000    38 ;
0000    39 ;
0000    40 ; AUTHOR: Scott G. Davis, 25-Mar-1983
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;
0000    44 ;
0000    45 ;**
0000    46 ;--
```

```
                     0000     48          .SBTTL  DECLARATIONS
                     0000     49 ;
                     0000     50 ; DEFAULT ADDRESSING MODE
                     0000     51 ;
                     0000     52          .DEFAULT DISPLACEMENT WORD
                     0000     53 ;
                     0000     54 ; INCLUDE FILES:
                     0000     55 ;
                     0000     56          $NFBDEF             ; Network function block definitions
                     0000     57          $NMADEF             ; Network management definitions
                     0000     58 ;
                     0000     59 ; MACROS:
                     0000     60 ;
                     0000     61          .MACRO  ONERROR ACTION,?L
                     0000     62          BLBS    R0,L
                     0000     63          ACTION
                     0000     64          L:
                     0000     65          .ENDM   ONERROR
                     0000     66 ;
                     0000     67 ; EQUATED SYMBOLS:
                     0000     68 ;
0000000A             0000     69 LF       =       10
0000000D             0000     70 CR       =       13
                     0000     71
                     0000     72 ; ***NOTE  References to PDBs assume they are of the form:
                     0000     73 ;
                     0000     74 ;        .BYTE   presence-flag
                     0000     75 ;        .BLKB   n           - data
                     0000     76 ;
                     0000     77
                     0000     78 ;
                     0000     79 ; OWN STORAGE:
                     0000     80 ;
                 00000000     81          .PSECT  CONCAR$PURE,NOWRT,NOEXE
                     0000     82          .SBTTL  READ ONLY DATA
                     0000     83
                 00000023'    0000     84 NOT_NI_MSG:      .LONG   20$-10$         ; Error message
                 00000008'    0004     85                  .LONG   10$
76 20 6E 6F 69 74 61 72 65 70 4F 0A   0008     86 10$:     .ASCII  <LF>/Operation valid only on Ethernet/<CR><LF>
6E 6F 20 79 6C 6E 6F 20 64 69 6C 61   0014
   0A 0D 74 65 6E 72 65 68 74 45 20   0020
                     002B     87 20$:
                     002B     88
3A 30 54 45 4E 5F 00000033'010E0000'  002B     89 NET_DESC:        .ASCID  /_NET0:/
                     0039     90
```

```
                0039     92          .SBTTL  READ/WRITE DATA
            00000000     93          .PSECT  CONCAR$IMPURE,NOEXE,WRT,LONG
                0000     94
00000008        0000     95 IOSB:    .BLKQ   1                       ; I/O status block
                0008     96
00000010        0008     97 CIRC_DESC:       .BLKQ   1               ; Descriptor to pass to $ASSIGN
00000014        0010     98 NICE_MSG_END:    .BLKL   1               ; Pointer to end of message
                0014     99
00000016        0014    100 NETCHAN:        .BLKW   1               ; Network I/O channel
```

```
                                                .PSECT  CONCAR$CODE,NOWRT
                                        ;++
                                        ;
                                        ; CON$PARSE_NICE
                                        ;
                                        ; This routine takes a parameter ID, locates it in a NICE message, and
                                        ; parses the data associated with the parameter.
                                        ;
                                        ; CALLING SEQUENCE:
                                        ;
                                        ;       CALLS   #4,CON$PARSE_NICE
                                        ;
                                        ; INPUT PARAMETERS:
                                        ;
                                        ;       R1,R2   - Message descriptor
                                        ;       4(AP)   - Parameter type to match
                                        ;       8(AP)   - Address to store data, if parameters match
                                        ;
                                        ; IMPLICIT INPUTS:
                                        ;
                                        ;       NICE response message for a NODE, of the form:
                                        ;
                                        ;               .BYTE   1
                                        ;               .WORD   error detail
                                        ;               .BYTE   NMA$C_ENT_NOD
                                        ;               .WORD   node address
                                        ;               .ASCIC  node name
                                        ;               .REPT   N
                                        ;               .WORD   PARAM_ID (DATA ID from network management spec)
                                        ;               .WORD   PARAM_TYPE (DATA TYPE from network management spec)
                                        ;               data
                                        ;               .ENDR
                                        ;
                                        ; OUTPUT PARAMETERS:
                                        ;
                                        ;       R0 - 0==> parameter not found, 1==>parameter found
                                        ;
                                        ; IMPLICIT OUTPUTS:
                                        ;
                                        ;       If parameter is in message, it is stored at the address provided,
                                        ; in the format that is appropriate, e.g. counted ASCII string.
                                        ;
                                        ; COMPLETION CODES:
                                        ;
                                        ; SIDE EFFECTS:
                                        ;
                                        ;       None
                                        ;
                                        ;--
                                        ;
                OFFC                    CON$PARSE_NICE::        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          50    D4                      CLRL    R0                      ; Assume parameter is not found
0010'CF   52    51    C1                ADDL3   R1,R2,NICE_MSG_END      ; Store pointer to end of message
          52    03    C0                ADDL2   #3,R2                   ; Skip response code and error detail
                                        ;
                                        ;       ASSUME  NMA$C_ENT_NOD   EQ 0
                                        ;
```

```
                82   95   000D   159          TSTB    (R2)+                   ; Check for node entity
                3B   12   000F   160          BNEQ    100$                    ; If NEQ bogus msg
0000'CF         82   B0   0011   161          MOVW    (R2)+,CON$W_ADDR_DNA    ; Store the address
56    0000'CF   9E   0016   162          MOVAB   CON$T_SHOW_NODE,R6      ; Point to name
      53        82   9A   001B   163          MOVZBL  (R2)+,R3                ; Get length of node name
      86        53   90   001E   164          MOVB    R3,(R6)+                ; Store name length
                51   DD   0021   165          PUSHL   R1                      ; Save original message lth
66    62        53   28   0023   166          MOVC3   R3,(R2),(R6)            ; Store name
      52        51   D0   0027   167          MOVL    R1,R2                   ; Reset message pointer
                51 8ED0   002A   168          POPL    R1                      ; Restore message length
                         002D   169 ;
                         002D   170 ; Main parsing loop
                         002D   171 ;
0010'CF         52   D1   002D   172 20$:     CMPL    R2,NICE_MSG_END         ; At end of message?
                18   1E   0032   173          BGEQU   100$                    ; If GEQU yes - param not found
                5B   D4   0034   174          CLRL    R11                     ; Don't store unless there is a match
  04 AC         82   B1   0036   175          CMPW    (R2)+,4(AP)             ; Is this the param?
                04   12   003A   176          BNEQ    30$                     ; If NEQ no - just skip over
  5B    08 AC   D0   003C   177          MOVL    8(AP),R11               ; Need to store param - get its address
004D'CF         00   FB   0040   178 30$:     CALLS   #0,DECODE_TYPE          ; Decode the parameter
                5B   D5   0045   179          TSTL    R11                     ; Was data stored?
                E4   13   0047   180          BEQL    20$                     ; If EQL no - keep looking
      50        01   D0   0049   181          MOVL    #1,R0                   ; Parameter found
                04   004C   182 100$:     RET                             ; Done
```

```
                     004D   184 ;++
                     004D   185 ;
                     004D   186 ; DECODE_TYPE - decode NICE message data type, possibly storing data
                     004D   187 ;
                     004D   188 ; This routine may be called recursively, for coded-mltiples
                     004D   189 ;
                     004D   190 ; CALLING SEQUENCE
                     004D   191 ;
                     004D   192 ;       CALLS   #0,DECODE_TYPE
                     004D   193 ; INPUTS:
                     004D   194 ;
                     004D   195 ;       R2 - points to param type in message
                     004D   196 ;       R11 - points to data storage area; 0==>just skip m    age field
                     004D   197 ;
                     004D   198 ; OUTPUTS:
                     004D   199 ;
                     004D   200 ;       Data may be stored
                     004D   201 ;
                     004D   202 ;--
                     004D   203
             0600    004D   204 DECODE_TYPE:    .WORD   ^M<R9,R10>
    5A   82   9A     004F   205                 MOVZBL  (R2)+,R10               ; Get the param type
    59   5A   D0     0052   206                 MOVL    R10,R9                  ; Make a copy
5A   C0 8F   8A     0055   207                 BICB    #^C<NMA$M_PTY_CLE>,R10  ; Get no. of fields or bytes
0F 59   07   E1     0059   208                 BBC     #NMA$V_PTY_COD,R9,100$  ; If BC not encoded
21 59   06   E1     005D   209                 BBC     #NMA$V_PTY_MUL,R9,175$  ; If BC not coded-multiple
        5A   DD     0061   210                 PUSHL   R10                     ; Save recursion count
E6 AF   00   FB     0063   211 10$:     CALLS   #0,DECODE_TYPE          ; Make recursive call
     F9 6E   F5     0067   212                 SOBGTR  (SP),10$                ; Loop
        25   11     006A   213                 BRB     200$                    ; Common exit
                     006C   214 ;
                     006C   215 ; Type is not coded
                     006C   216 ;
06 59   06   E0     006C   217 100$:    BBS     #NMA$V_PTY_ASC,R9,150$  ; If BS Image ASCII data
5A   F0 8F   8A     0070   218                 BICB    #^C<NMA$M_PTY_NLE>,R10  ; Get no. of characters of data
        0C   12     0074   219                 BNEQ    175$                    ; If NEQ not image hex
   5A   82   9A     0076   220 150$:    MOVZBL  (R2)+,R10               ; Get lth of image data
        5B   D5     0079   221                 TSTL    R11                     ; Storing?
        09   13     007B   222                 BEQL    180$                    ; If EQL no
   8B   5A   90     007D   223                 MOVB    R10,(R11)+              ; Store lth of string
        09   11     0080   224                 BRB     190$                    ; Go to common code
                     0082   225
                     0082   226 ;
                     0082   227 ; Type is coded, but not multiple
                     0082   228 ;
        5B   D5     0082   229 175$:    TSTL    R11                     ; Storing?
        05   12     0084   230                 BNEQ    190$                    ; If NEQ yes
   52   5A   C0     0086   231 180$:    ADDL2   R10,R2                  ; Update message pointer
        06   11     0089   232                 BRB     200$                    ; Done
   8B   82   90     008B   233 190$:    MOVB    (R2)+,(R11)+            ; Store a byte
     FA 5A   F5     008E   234                 SOBGTR  R10,190$                ; Loop
                     0091   235
             04      0091   236 200$:    RET                             ; Done
```

```
                     0092   238 ;++
                     0092   239 ;
                     0092   240 ; CON$GET_SLI - translate service circuit into VMS device name
                     0092   241 ;
                     0092   242 ; INPUTS:
                     0092   243 ;
                     0092   244 ;       Circuit name in .ASCIC in PDB$G_CON_SLI
                     0092   245 ;
                     0092   246 ;--
                     0092   247
                     0092   248 CON$GET_SLI::
                     0092   249         $ASSIGN_S DEVNAM = NET_DESC, CHAN = NETCHAN
                     00A3   250         ONERROR RET                     ; If error done
                     00A7   251
    56  0000'CF  9E  00A7   252         MOVAB   CON$NFB,R6              ; Get address of NFB
          22 90  00AC   253         MOVB    #NFB$C_FC_SHOW,-        ; This is a show function
          66             00AE   254                 NFB$B_FCT(R6)
          04 90  00AF   255         MOVB    #NFB$C_DB_CRI,-         ; Circuit database
        02 A6            00B1   256                 NFB$B_DATABASE(R6)
          00 90  00B3   257         MOVB    #NFB$C_OP_EQL,-         ; Match criterion
        03 A6            00B5   258                 NFB$B_OPER(R6)
 04020041 8F  D0  00B7   259         MOVL    #NFB$C_CRI_NAM,-        ; Match on circuit name
          04 A6          00BD   260                 NFB$L_SRCH_KEY(R6)
 04010020 8F  D0  00BF   261         MOVL    #NFB$C_CRI_TYP,-        ; I want the device type back
          10 A6          00C5   262                 NFB$L_FLDID(R6)
 04020042 8F  D0  00C7   263         MOVL    #NFB$C_CRI_VMSNAM,-     ; I want the VMS device name back
          14 A6          00CD   264                 NFB$L_FLDID+4(R6)
    50  0001'CF  9E  00CF   265         MOVAB   PDB$G_CON_SLI+1,R0      ; Point to circuit name
    53  0004'CF  9E  00D4   266         MOVAB   CON$P2BUF+4,R3          ; Point to search key storage
        83   80  9B  00D9   267         MOVZBW  (R0)+,(R3)+             ; Store length of name
63  60   FE A3  28  00DC   268         MOVC3   -2(R3),(R0),(R3)        ; Store the name
                     00E1   269         $QIOW_S FUNC=#IO$_ACPCONTROL,-  ; Issue the control function
                     00E1   270                 CHAN=NETCHAN,-
                     00E1   271                 IOSB=IOSB,-
                     00E1   272                 P1=CON$NFB_DESC,-
                     00E1   273                 P2=#CON$P2BUF_DESC,-
                     00E1   274                 P4=#CON$P4BUF_DESC
                     010E   275         ONERROR RET
                     0112   276
    50  0000'CF  D0  0112   277         MOVL    IOSB,R0                 ; Get IOSB
0000'8F  50  B1  0117   278         CMPW    R0,#SS$_ENDOFFILE       ; Legitimate device?
          05 12  011C   279         BNEQ    5$                      ; If NEQ maybe
    50  0000'8F  B0  011E   280         MOVW    #SS$_NOSUCHDEV,R0       ; Return this error
                     0123   281 5$:     ONERROR RET                     ; If error, so long
                     0127   282
    50  0000'CF  9E  0127   283         MOVAB   CON$P4BUF,R0            ; Get address of response buffer
        06   80  D1  012C   284         CMPL    (R0)+,#NMA$C_CIRTY_NI   ; Is this an Ethernet circuit?
             0E  13  012F   285         BEQL    10$                     ; If EQL yes
    53  0004'CF  D0  0131   286         MOVL    NOT_NI_MSG+4,R3         ; Address of message
    52  0000'CF  D0  0136   287         MOVL    NOT_NI_MSG,R2           ; Length of message
           FEC2' 30  013B   288         BSBW    NCP$TTY_WRITE           ; Output the message
                 04  013E   289         RET                             ; That is all!
                     013F   290
    51  0008'CF  9E  013F   291 10$:    MOVAB   CIRC_DESC,R1            ; Point to descriptor area
        81   80  3C  0144   292         MOVZWL  (R0)+,(R1)+             ; Get the length of the name
        61   50  D0  0147   293         MOVL    R0,(R1)                 ; Get its address
0000'CF  71  DE  014A   294         MOVAL   -(R1),CON$XE_DESCADDR   ; Move to desc addr to common place
```

```
 05  014F   295 100$:   RSB                                 ; Done
     0150   296
     0150   297         .END
```

```
$$T1                            = 00000001
CIRC_DESC                         00000008 R      03
CON$GET_SLI                       00000092 RG     04
CON$NFB                           ******** *X     04
CON$NFB_DESC                      ******** *X     04
CON$P2BUF                         ******** *X     04
CON$P2BUF_DESC                    ******** *X     04
CON$P4BUF                         ******** *X     04
CON$P4BUF_DESC                    ******** *X     04
CON$PARSE_NICE                    00000000 RG     04
CON$T_SHOW_NODE                   ******** *X     04
CON$W_ADDR_DNA                    ******** *X     04
CON$XE_DESCADDR                   ******** *X     04
CR                              = 0000000D
DECODE_TYPE                       0000004D R      04
IO$_ACPCONTROL                    ******** *X     04
IOSB                              00000000 R      03
LF                              = 0000000A
NCP$TTY_WRITE                     ******** *X     04
NETCHAN                           00000014 R      03
NET_DESC                          0000002B R      02
NFB$B_DATABASE                  = 00000002
NFB$B_FCT                       = 00000000
NFB$B_OPER                      = 00000003
NFB$C_CRI_NAM                   = 04020041
NFB$C_CRI_TYP                   = 04010020
NFB$C_CRI_VMSNAM                = 04020042
NFB$C_DB_CRI                    = 00000004
NFB$C_FC_SHOW                   = 00000022
NFB$C_OP_EQL                    = 00000000
NFB$L_FLDID                     = 00000010
NFB$L_SRCH_KEY                  = 00000004
NICE_MSG_END                      00000010 R      03
NMA$C_CIRTY_NI                  = 00000006
NMA$C_ENT_NOD                   = 00000000
NMA$M_PTY_CLE                   = 0000003F
NMA$M_PTY_NLE                   = 0000000F
NMA$V_PTY_ASC                   = 00000006
NMA$V_PTY_COD                   = 00000007
NMA$V_PTY_MUL                   = 00000006
NOT_NI_MSG                        00000000 R      02
PDB$G_CON_SLI                     ******** *X     04
SS$_ENDOFFILE                     ******** *X     04
SS$_NOSUCHDEV                     ******** *X     04
SYS$ASSIGN                        ******** GX     04
SYS$QIOW                          ******** GX     04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
CONCAR$PURE	00000039 (57.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
CONCAR$IMPURE	00000016 (22.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG

```
CONCAR$CODE                     00000150 (  336.) 04 (  4.) NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.04	00:00:01.31
Command processing	124	00:00:00.73	00:00:06.78
Pass 1	408	00:00:12.57	00:00:34.79
Symbol table sort	0	00:00:01.82	00:00:05.63
Pass 2	65	00:00:01.99	00:00:05.30
Symbol table output	7	00:00:00.08	00:00:00.12
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	640	00:00:17.26	00:00:53.96

```
The working set limit was 1650 pages.
67147 bytes (132 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1394 non-local and 18 local symbols.
297 source lines were read in Pass 1, producing 18 object records in Pass 2.
17 pages of virtual memory were used to define 16 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NET.MLB;1	1
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	12

```
1505 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NCPCONMAN/OBJ=OBJ$:NCPCONMAN MSRC$:NCPCONMAN/UPDATE=(ENH$:NCPCONMAN)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB+SHRLIB$:NET/LIB
```

NCPCONCAR
LIS

NCPERRMSG
LIS

NCPCONMAN
LIS

NCPLIBRY
B32

NCPMAIN
LIS

NCPNETIO
LIS

NMAHEAD
B32

NCPLIBRY
LIS

NMATAIL
B32