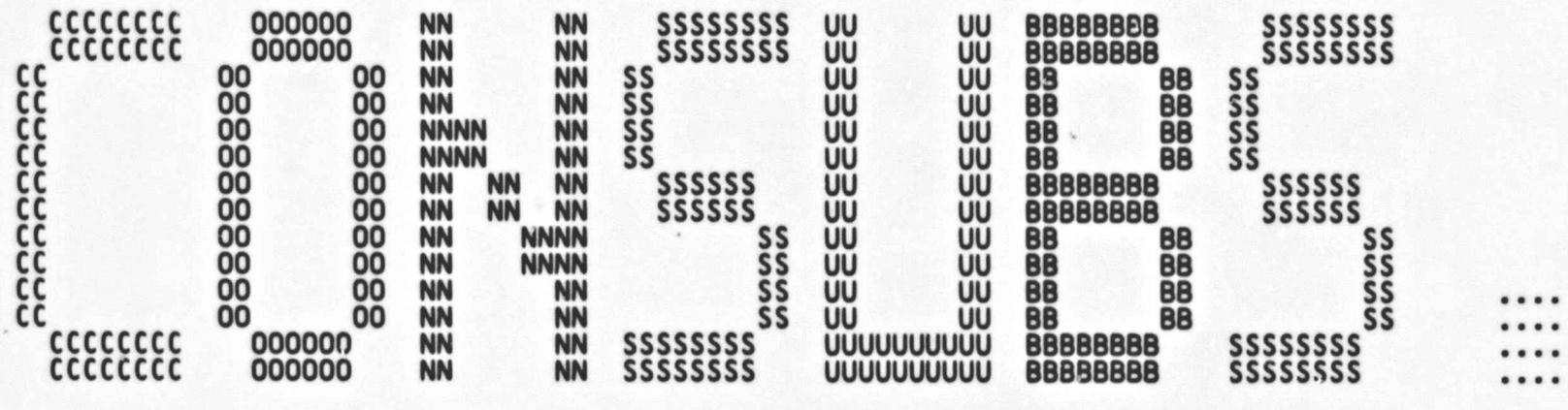
```
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
```

FILEID**CONSUBS



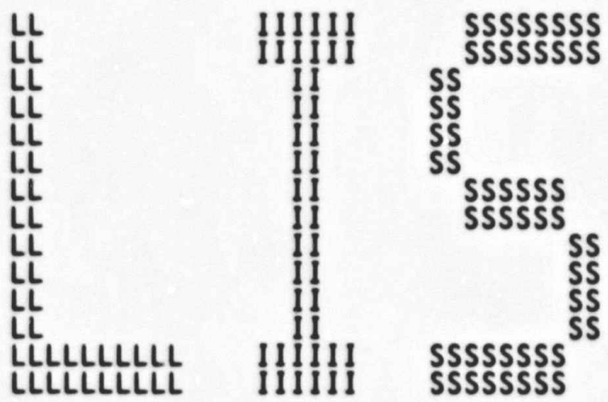


Table of contents

```
0000     1          .TITLE  CONSUBS - Connection Management Utility Subroutines
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27
0000    28 ;++
0000    29 ; FACILITY: EXECUTIVE, CLUSTER MANAGEMENT
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;       This module contains several utility routines for the connection manager.
0000    33 ;
0000    34 ; ENVIRONMENT: VAX/VMS
0000    35 ;
0000    36 ; AUTHOR: Dave Thiel,    CREATION DATE: 13-Dec-1983
0000    37 ;
0000    38 ; MODIFIED BY:
0000    39 ;
0000    40 ;       V03-003 DWT0198         David W. Thiel                  23-Mar-1984
0000    41 ;               Add CNX$RESOURCE_CHECK, CNX$RESOURCE_FAIL, CNX$RESOURCE_SUCC
0000    42 ;               to manage system resource exhaustion.
0000    43 ;
0000    44 ;       V03-002 DWT0187         David W. Thiel                  5-Mar-1984
0000    45 ;               Really repair random number generation.
0000    46 ;
0000    47 ;       V03-001 DWT0176         David W. Thiel                  21-Feb-1984
0000    48 ;               Correct random number generation.
0000    49 ;
0000    50 ;--
0000    51
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                        $CLUBDEF                        ; CLUster Block offsets
                        $CSBDEF                         ; CSB Offsets
                        $DYNDEF                         ; Data structure type codes
                        $FKBDEF                         ; Fork block offsets
                        $SBDEF                          ; System block offsets

                ;*****************************************************************************
                ;
                ; NOTE: The following assumptions are in effect for this entire module.
                ;
                ;*****************************************************************************

                        .DEFAULT        DISPLACEMENT,WORD

00000000                .PSECT  $$$100,LONG             ; PSECT for code
```

```
                                0000     73          .SBTTL  CNX$ALLOZMEM - Allocate and zero memory
                                0000     74
                                0000     75 ;++
                                0000     76 ;
                                0000     77 ; FUNCTIONAL DESCRIPTION:
                                0000     78 ;
                                0000     79 ;       This routine allocates and zeroes a piece of pool.  The size of
                                0000     80 ;       the allocated block is stored in the block.  The type field is set
                                0000     81 ;       to DYN$C_CLU.
                                0000     82 ;
                                0000     83 ; CALLING SEQUENCE:
                                0000     84 ;
                                0000     85 ;       JSB     CNX$ALLOZMEM
                                0000     86 ;
                                0000     87 ; INPUT PARAMETERS:
                                0000     88 ;
                                0000     89 ;       R1:     Size of the block of memory to allocate.
                                0000     90 ;
                                0000     91 ; OUTPUT PARAMETERS:
                                0000     92 ;
                                0000     93 ;       R2:     Address of the allocated block of pool.
                                0000     94 ;
                                0000     95 ; COMPLETION CODES:
                                0000     96 ;
                                0000     97 ;       R0 contains status.
                                0000     98 ;
                                0000     99 ; SIDE EFFECTS:
                                0000    100 ;
                                0000    101 ;       NONE
                                0000    102 ;
                                0000    103 ;--
                                0000    104
                                0000    105 CNX$ALLOZMEM::
                                0000    105 CNX$ALLOZMEM::
 00000000'GF   16               0000    106          JSB     G^EXE$ALONONPAGED       ; Allocate memory
          16   50   E9          0006    107          BLBC    R0,10$                  ; Branch on error
               3E   BB          0009    108          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save volatile registers
62   51   00   6E   00          2C      000B    109          MOVC5   #0,(SP),#0,R1,(R2)      ; Zero the block
               3E   BA          0011    110          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore registers
        08 A2   51   B0          0013    111          MOVW    R1,FKB$W_SIZE(R2)       ; Block size
     0A A2   65   8F   90        0017    112          MOVB    #DYN$C_CLU, -           ; Store generic cluster type
                                 001C    113                  FKB$B_TYPE(R2)
          50   00'  D0          001C    114          MOVL    S^#SS$_NORMAL,R0        ; Set success status
               05   001F    115 10$:     RSB                                     ; Return
                    0020    116
```

```
                     0020  118          .SBTTL  CNX$RANDOM - Generate a random number
                     0020  119
                     0020  120 ;++
                     0020  121 ;
                     0020  122 ; FUNCTIONAL DESCRIPTION:
                     0020  123 ;
                     0020  124 ;       This routine generate a random integer whose range is determined by R0.
                     0020  125 ;
                     0020  126 ; CALLING SEQUENCE:
                     0020  127 ;
                     0020  128 ;       JSB     CNX$RANDOM
                     0020  129 ;
                     0020  130 ; INPUT PARAMETERS:
                     0020  131 ;
                     0020  132 ;       R4:     Address of CLUB
                     0020  133 ;       R0:     Upper bound+1 on number generated (assumed positive)
                     0020  134 ;
                     0020  135 ; OUTPUT PARAMETERS:
                     0020  136 ;
                     0020  137 ;       R0:     Random integer in range 0..R0
                     0020  138 ;
                     0020  139 ; COMPLETION CODES:
                     0020  140 ;
                     0020  141 ;       NONE
                     0020  142 ;
                     0020  143 ; SIDE EFFECTS:
                     0020  144 ;
                     0020  145 ;       R1 is destoryed.
                     0020  146 ;
                     0020  147 ;--
                     0020  148
                     0020  149 CNX$RANDOM::
                50   DD  0020  150          PUSHL   R0                      ; Save upper bound
01   00010DCD 8F     00B0 C4  7A  0022  151          EMUL    CLUB$L_RANDOM(R4), -    ; Compute new context using
                50           0026
                             002D  152                  #69069,#1,R0            ;   context = context*69069 + 1
                     00B0 C4  50  D0  002D  153          MOVL    R0,CLUB$L_RANDOM(R4)    ; Update context
                          6E  51  D1  0032  154          CMPL    R1,(SP)                 ; Test for overflow in EDIV
                              05  1F  0035  155          BLSSU   10$                     ; Branch if no overflow and positive
                          50  51  CC  0037  156          XORL2   R1,R0                   ; Mash it all into R0
                              51  D4  003A  157          CLRL    R1
                50   51  50  8E  7B  003C  158 10$:     EDIV    (SP)+,R0,R1,R0          ; Remainder to R0
                              05  0041  159          RSB                             ; Return number in R0, range 0..R0-1
                             0042  160
```

```
                     0042   162          .SBTTL  CNX$RANDOM_INIT - Initialize Random Number Generator Context
                     0042   163
                     0042   164 ;++
                     0042   165 ;
                     0042   166 ; FUNCTIONAL DESCRIPTION:
                     0042   167 ;
                     0042   168 ;       This routine initializes the random number generator context
                     0042   169 ;
                     0042   170 ; CALLING SEQUENCE:
                     0042   171 ;
                     0042   172 ;       JSB     CNX$RANDOM_INIT
                     0042   173 ;
                     0042   174 ; INPUT PARAMETERS:
                     0042   175 ;
                     0042   176 ;       R4:     Address of CLUB
                     0042   177 ;
                     0042   178 ; OUTPUT PARAMETERS:
                     0042   179 ;
                     0042   180 ;       NONE
                     0042   181 ;
                     0042   182 ; COMPLETION CODES:
                     0042   183 ;
                     0042   184 ;       NONE
                     0042   185 ;
                     0042   186 ; SIDE EFFECTS:
                     0042   187 ;
                     0042   188 ;       R0 and R1 are destoryed.
                     0042   189 ;
                     0042   190 ;--
                     0042   191
                     0042   192 CNX$RANDOM_INIT::
 50   00000000'GF    7D   0042   193          MOVQ    G^SCS$GB_SYSTEMID,R0    ; Get local system ID
      50    51       AC   0049   194          XORW2   R1,R0                   ; Mash into one longword
 50   00000002'GF    CC   004C   195          XORL2   G^EXE$GQ_SYSTIME+2,R0   ; Incorporate active part of time
      51    FD AF    9E   0053   196 10$:     MOVAB   B^10$,R1                ; Get another system dependent value
00B0 C4    50    51   CD   0057   197          XORL3   R1,R0, -                ; Merge it into value
                                   005D   198                  CLUB$L_RANDOM(R4)       ; and store it
                           05      005D   199          RSB
                                   005E   200
```

```
                                005E    202          .SBTTL  CNX$RESOURCE_SUCC - Note resouce allocation success
                                005E    203          .SBTTL  CNX$RESOURCE_FAIL - Check for resource exhaustion following failure
                                005E    204          .SBTTL  CNX$RESOURCE_CHECK - Check for resource exhaustion
                                005E    205
                                005E    206 ;++
                                005E    207 ;
                                005E    208 ; FUNCTIONAL DESCRIPTION:
                                005E    209 ;
                                005E    210 ;       These routines attempt to prevent infinite resource waiting loops in the
                                005E    211 ;       connection manager and other code critical to the cluster.
                                005E    212 ;       Calls to these routines may be placed in resource wait loops with the
                                005E    213 ;       following constraint:  If the routines are ever called on a failure to
                                005E    214 ;       allocate a particular resource, they must also be called when that resource
                                005E    215 ;       is successfully allocated.  If the caller "bails-out" of the wait loop,
                                005E    216 ;       these routines must be called indicating a successfull allocation.  If
                                005E    217 ;       this is not done, false resource exhausted conditions will be detected
                                005E    218 ;       and machines will unnecessarily be shut down.
                                005E    219 ;
                                005E    220 ; CALLING SEQUENCE:
                                005E    221 ;
                                005E    222 ;       JSB     CNX$RESOURCE_CHECK - Resource allocation success/failure status in R
                                005E    223 ;       JSB     CNX$RESOURCE_FAIL - Implied resource allocation failure
                                005E    224 ;       JSB     CNX$RESOURCE_SUCC - Implied resource allocation success
                                005E    225 ;
                                005E    226 ; INPUT PARAMETERS:
                                005E    227 ;
                                005E    228 ;       R0:     Resource allocation success/failure (CNX$RESOURCE_CHECK only)
                                005E    229 ;
                                005E    230 ; OUTPUT PARAMETERS:
                                005E    231 ;
                                005E    232 ;       NONE
                                005E    233 ;
                                005E    234 ; COMPLETION CODES:
                                005E    235 ;
                                005E    236 ;       NONE
                                005E    237 ;
                                005E    238 ; SIDE EFFECTS:
                                005E    239 ;
                                005E    240 ;       All registers are preserved.
                                005E    241 ;       If a state of resource exhaustion is discovered, the system will
                                005E    242 ;       shut itself down with a resources exhausted bugcheck.
                                005E    243 ;
                                005E    244 ;--
                                005E    245
                                005E    246 CNX$RESOURCE_CHECK::
                OF 50   E9      005E    247          BLBC    R0,CNX$RESOURCE_FAIL    ; Handle failure case
                                0061    248 CNX$RESOURCE_SUCC::
                  54    DD      0061    249          PUSHL   R4                      ; Save register
    54  00000000'GF    D0      0063    250          MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
                6C A4   D4      006A    251          CLRL    CLUB$L_RETRYCNT(R4)     ; Clear timeout cell
                  10    BA      006D    252          POPR    #^M<R4>                 ; Restore register
                        05      006F    253          RSB
                                0070    254
                                0070    255 CNX$RESOURCE_FAIL::
                  11    BB      0070    256          PUSHR   #^M<R0,R4>              ; Save register
    54  00000000'GF    D0      0072    257          MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
                6C A4   D5      0079    258          TSTL    CLUB$L_RETRYCNT(R4)     ; Is ths the first failure?
```

```
                        10   12  007C   259          BNEQ    10$                             ; Branch if not first failure
              50 00000000'GF   3C  007E   260          MOVZWL  G^CLU$GW_RECNXINT,R0            ; Retry interval
     6C A4    00000000'GF   50  C1  0085   261          ADDL3   R0,G^EXE$GL_ABSTIM, -           ; Compute time-out absolute time
                                  008E   262                  CLUB$L_RETRYCNT(R4)
          6C A4 00000000'GF   D1  008E   263 10$:       CMPL    G^EXE$GL_ABSTIM, -              ; Compare current time to time-out
                                  0096   264                  CLUB$L_RETRYCNT(R4)
                        03   1A  0096   265          BGTRU   20$                             ; Branch if timed-out
                        11   BA  0098   266          POPR    #^M<R0,R4>                      ; Restore register
                        05   009A   267              RSB
                             009B   268
                             009B   269 20$:       BUG_CHECK       RESEXH,FATAL            ; Resources exhausted, node exiting cluster
                             009F   270
                             009F   271          .END
```

```
BUG$_RESEXH                     ********    X   02
CLU$GL_CLUB                     ********    X   02
CLU$GW_RECNXINT                 ********    X   02
CLUB$L_RANDOM                 = 000000B0
CLUB$L_RETRYCNT               = 0000006C
CNX$ALLOCMEM                    00000000 RG     02
CNX$RANDOM                      00000020 RG     02
CNX$RANDOM_INIT                 00000042 RG     02
CNX$RESOURCE_CHECK              0000005E RG     02
CNX$RESOURCE_FAIL               00000070 RG     02
CNX$RESOURCE_SUCC               00000061 RG     02
DYN$C_CLU                     = 00000065
EXE$AL_NONONPAGED               ********    X   02
EXE$GL_ABSTIM                   ********    X   02
EXE$GQ_SYSTIME                  ********    X   02
FKB$B_TYPE                    = 0000C00A
FKB$W_SIZE                    = 00000008
SCS$GB_SYSTEMID                 ********    X   02
SS$_NORMAL                      ********    X   02
```

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
$$$100	0000009F (159.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	41	00:00:00.07	00:00:02.02
Command processing	128	00:00:00.45	00:00:03.11
Pass 1	203	00:00:03.04	00:00:17.72
Symbol table sort	0	00:00:00.44	00:00:02.17
Pass 2	60	00:00:00.68	00:00:02.73
Symbol table output	4	00:00:00.02	00:00:00.03
Psect synopsis output	1	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	439	00:00:04.71	00:00:27.80

```
The working set limit was 1350 pages.
24460 bytes (48 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 447 non-local and 5 local symbols.
271 source lines were read in Pass 1, producing 13 object records in Pass 2.
13 pages of virtual memory were used to define 12 macros.
```

J 6

```
                    +---------------------------+
                    ! Macro library statistics !
                    +---------------------------+

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1                        0
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                               6
_$255$DUA28:[SYSLIB]STARLET.MLB;2                            3
TOTALS (all libraries)                                       9

513 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:CONSUBS/OBJ=OBJ$:CONSUBS MSRC$:CONSUBS/UPDATE=(ENH$:CONSUBS)+EXECML$/LIB+LIB$:CLUSTER/LIB
```

0393 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPCALL.
LIS

CSPCJFMAS.
LIS

CSP
LIS

CSPBRKTHR.
LIS

CONUTIL.
LIS

CSPCALLACT.
LIS

CONSUBS.
LIS