```
LLL               BBBBBBBBBBBB      RRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RR           RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBB          BBB  RRR          RRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBBBBBBBBBBBB     RRRRRRRRRRRRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR   RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLL               BBB          BBB  RRR      RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBB      RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
LLLLLLLLLLLLLLLL  BBBBBBBBBBBBB     RRR         RRR
```

```
_$2

Val
---
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
002
00F
00F
00F
00F
00F
00F
00F
00F
00F
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
  CCCCCCCC    AAAAAA      CCCCCCCC  HH      HH  EEEEEEEEEE
  CCCCCCCC    AAAAAA      CCCCCCCC  HH      HH  EEEEEEEEEE
CC           AA      AA  CC        HH      HH  EE
CC           AA      AA  CC        HH      HH  EE
CC           AA      AA  CC        HH      HH  EE
CC           AA      AA  CC        HH      HH  EE
CC           AA      AA  CC        HHHHHHHHHH  EEEEEEEE
CC           AA      AA  CC        HHHHHHHHHH  EEEEEEEE
CC           AAAAAAAAAA  CC        HH      HH  EE
CC           AAAAAAAAAA  CC        HH      HH  EE
CC           AA      AA  CC        HH      HH  EE              ....
CC           AA      AA  CC        HH      HH  EE              ....
  CCCCCCCC   AA      AA    CCCCCCCC  HH      HH  EEEEEEEEEE      ....
  CCCCCCCC   AA      AA    CCCCCCCC  HH      HH  EEEEEEEEEE      ....

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  cache   Disk block cache routines
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 ;   FACILITY
0000    29 ;
0000    30 ;       Native mode librarian
0000    31 ;
0000    32 ;   ABSTRACT
0000    33 ;
0000    34 ;       Routines to manipulate the disk block cache
0000    35 ;       in dynamic memory.
0000    36 ;
0000    37 ;   ENVIRONMENT
0000    38 ;
0000    39 ;       Native mode, user mode
0000    40 ;
0000    41 ;   AUTHOR
0000    42 ;
0000    43 ;       Tim Halvorsen, Benn Schreiber, July 1979
0000    44 ;
0000    45 ;   MODIFIED BY
0000    46 ;
0000    47 ;       V03-001         CWH0001         CW Hobbs        14-Aug-1982
0000    48 ;               Change name of $LBRCTLTBL macro to $LBRCTLDEF so that
0000    49 ;               name produced by SDL matches.
0000    50 ;
0000    51 ;       V02-008         RPG0018         Bob Grosso      22-Dec-1981
0000    52 ;               Support deallocation of cache header entries.
0000    53 ;
0000    54 ;       V02-007         RPG0017         Bob Grosso      02-Dec-1981
0000    55 ;               Support allocation of cache header entries from
0000    56 ;               common block.
0000    57 ;
```

```
0000     58 ;       V02-003         RPG0014         Bob Grosso      24-Jun-1981
0000     59 ;               Check status return from allocate in add_cache.
0000     60 ;
0000     61 ;       V02-002         RPG0013         Bob Grosso      19-Jan-1981
0000     62 ;               Record the highest VBN in the cache at ctx$l_hivbn.
0000     63 ;
0000     64 ;--
```

```
                        .sbttl  Declarations
                ;
                ;       Symbol defintions
                ;

                        $cachedef               ; Cache definitions
                        $ctxdef                 ; Context area definitions
                        $lbrdef                 ; Librarian definitions
                        $lbrctldef              ; Librarian control table

00000000                .psect  $code$,exe,nowrt

                        .default displacement,word
```

```
                     0000     80          .sbttl  add_cache, Add cache entry given VBN
                     0000     81 ;---
                     0000     82 ;
                     0000     83 ;       Add a cache entry to the block cache and return
                     0000     84 ;       the address to the caller.
                     0000     85 ;
                     0000     86 ; Inputs:
                     0000     87 ;
                     0000     88 ;       R0 = VBN of disk block
                     0000     89 ;       R1 = Address of longword to receive cache entry address
                     0000     90 ;
                     0000     91 ; Outputs:
                     0000     92 ;
                     0000     93 ;       None
                     0000     94 ;---
                     0000     95
                     0000     96 add_cache::
        3C   BB      0000     97          pushr   #^M<r2, r3, r4, r5>     ; save registers
        39   10      0002     98          bsbb    find_vbn_entry          ; look it up
     05 50   E9      0004     99          blbc    r0,40$                  ; branch if not found, go insert
        50   D4      0007    100          clrl    r0                      ; error if already in cache
        3C   BA      0009    101          popr    #^m<r2, r3, r4, r5>
             05      000B    102          rsb
                     000C    103 ;
                     000C    104 ; add to cache
                     000C    105 ;
  50    0E   D0      000C    106 40$:     movl    #cache$c_length,r0      ; length of block
        67   10      000F    107          bsbb    alloc_cache_hdr         ; allocate cache header entry
    26  50   E9      0011    108          blbc    r0,60$                  ; return error status
65      51   D0      0014    109          movl    r1,(r5)                 ; return address to caller
61      62   D0      0017    110          movl    cache$l_link(r2),cache$l_link(r1) ; and link into hash list
        62   51      D0 001A 111          movl    r1,cache$l_link(r2)
  04    A1   54      D0 001D 112          movl    r4,cache$l_vbn(r1)      ; set vbn into cache entry
52      0000'CF      D0 0021 113          movl    lbr$gl_control,r2       ; control table index
  52      0E A2      D0 0026 114          movl    lbr$l_ctxptr(r2),r2     ; context block address
  53      46 A2      D0 002A 115          movl    ctx$l_hivbn(r2),r3      ; highest vbn in cache
        53   54      D1 002E 116          cmpl    r4,r3                   ; compare highest to most recent vbn
             04      15 0031 117          bleq    50$
    46  A2   54      D0 0033 118          movl    r4,ctx$l_hivbn(r2)      ; save new highest vbn
        50   01      D0 0037 119 50$:     movl    #1,r0                   ; return success
             3C      BA 003A 120 60$:     popr    #^m<r2, r3, r4, r5>
                     05 003C 121          rsb
```

```
                                003D   123 ;++
                                003D   124 ;
                                003D   125 ;       Routine to look up vbn in the cache table
                                003D   126 ;
                                003D   127 ; Inputs:
                                003D   128 ;
                                003D   129 ;
                                003D   130 ; Outputs:
                                003D   131 ;
                                003D   132 ;       r0      0 not found
                                003D   133 ;               1 found
                                003D   134 ;       r1      pointer to entry (if found)
                                003D   135 ;       r2      pointer to previous entry
                                003D   136 ;
                                003D   137 ;--
                                003D   138 find_vbn_entry:
          54    50    7D      003D   139         movq    r0,r4                   ; put vbn in r4, return addr in r5
     51   0000'CF    D0      0040   140         movl    lbr$gl_control,r1       ; control table index
       51    0E A1    D0      0045   141         movl    lbr$l_ctxptr(r1),r1     ; context block address
       51    08 A1    D0      0049   142         movl    ctx$l_cache(r1),r1      ; point to cache hash table
     50    54    01    C3      004D   143         subl3   #1,r4,r0                ; hash function is vbn-1
50    FFFFFF80 8F    CA      0051   144         bicl2   #^C<<lbr$c_hashsize/4>-1>,r0 ; trim to table maximum
       52    6140    DE      0058   145         moval   (r1)[r0],r2             ; get address of previous
          51    62    D0      005C   146         movl    (r2),r1                 ; point to hash entry list
                14    13      005F   147 10$:     beql    30$                     ; if eql not in list
       04 A1    54    D1      0061   148         cmpl    r4,cache$l_vbn(r1)      ; is this the entry?
                0A    13      0065   149         beql    20$                     ; if eql yes
                0C    1A      0067   150         bgtru   30$                     ; if gtru no, and not in list
          52    51    D0      0069   151         movl    r1,r2                   ; bring up the rear
          51    61    D0      006C   152         movl    cache$l_link(r1),r1     ; and link to next
                EE    11      006F   153         brb     10$
50        01    D0      0071   154 20$:     movl    #1,r0
                05    0074   155         rsb
          50    D4    0075   156 30$:     clrl    r0
                05    0077   157         rsb
```

```
                                0078   159          .sbttl  alloc_cache_hdr, allocate dynamic memory for cache header
                                0078   160 ;---
                                0078   161 ;
                                0078   162 ;       alloc_cache_hdr
                                0078   163 ;
                                0078   164 ;       this routine allocates a cache header entry from a common block
                                0078   165 ;       of memory maintained by pointer in the context block.  If the
                                0078   166 ;       common block is not sufficient to meet the need then dynamic memory
                                0078   167 ;       is obtained by expanding the region, to increase the common block.
                                0078   168 ;
                                0078   169 ; inputs:
                                0078   170 ;
                                0078   171 ;       r0 = size of block to allocate
                                0078   172 ;
                                0078   173 ; outputs:
                                0078   174 ;
                                0078   175 ;       r1 = address of allocated block
                                0078   176 ;
                                0078   177 ;---
                                0078   178
                                0078   179 alloc_cache_hdr:
                      0C   BB   0078   180          pushr   #^m<r2,r3>
                                007A   181 ;
                                007A   182 ;       first check if common block is large enough
                                007A   183 ;
                52  0000'CF   D0   007A   184          movl    lbr$gl_control,r2       ; control table index
                  52   0E A2   D0   007F   185          movl    lbr$l_ctxptr(r2),r2     ; context block address
                  50   4A A2   D1   0083   186          cmpl    ctx$l_chdallsiz(r2),r0  ; see if common block is large enough
                       27   14   0087   187          bgtr    20$
                                0089   188 ;
                                0089   189 ;       if not large enough, then deallocate any fragment
                                0089   190 ;
                50   4A A2   D0   0089   191          movl    ctx$l_chdallsiz(r2),r0 ; size left over
                       08   13   008D   192          beql    10$
                51   4E A2   D0   008F   193          movl    ctx$l_chdalladr(r2),r1 ; address of left over
                  0000'CF   16   0093   194          jsb     dealloc_mem             ; deallocate left over
                                0097   195 ;
                                0097   196 ;       and allocate more
                                0097   197 ;
            50  00000E00 8F   D0   0097   198 10$:    movl    #<ctx$c_chdallblk*512>,r0 ; allocate new common block
                51   4E A2   9E   009E   199          movab   ctx$l_chdalladr(r2),r1  ; location to receive address
                     FF5B'   30   00A2   200          bsbw    get_zmem                ; Allocate the memory
                     1D 50   E9   00A5   201          blbc    r0,30$                  ; exit with error status
4A A2       00000E00 8F   D0   00A8   202          movl    #<ctx$c_chdallblk*512>,ctx$l_chdallsiz(r2) ; size of common block
                                00B0   203 ;
                                00B0   204 ;       obtain cache entry header from common block
                                00B0   205 ;
                   53   15   D0   00B0   206 20$:    movl    #<cache$c_length+7>,r3  ; round up to nearest
                   53   07   CA   00B3   207          bicl2   #7,r3                   ; multiple of four bytes
                4A A2   53   C2   00B6   208          subl2   r3,ctx$l_chdallsiz(r2)  ; decr block size
                51   4E A2   D0   00BA   209          movl    ctx$l_chdalladr(r2),r1  ; return address of cache header entry
                4E A2   53   C0   00BE   210          addl2   r3,ctx$l_chdalladr(r2)  ; remove from block
                   50   01   D0   00C2   211          movl    #1,r0                   ; return success
                        0C   BA   00C5   212 30$:     popr    #^m<r2,r3>
                             05   00C7   213          rsb
                                00C8   214
```

```
                     00C8   216          .sbttl  lookup_cache, Lookup a given VBN in the cache
                     00C8   217 ;---
                     00C8   218 ;
                     00C8   219 ;       Lookup a specified VBN in the disk block cache
                     00C8   220 ;       and return the address of the cache entry.
                     00C8   221 ;
                     00C8   222 ;   INPUTS:
                     00C8   223 ;
                     00C8   224 ;       R0 = VBN to lookup
                     00C8   225 ;       R1 = Address of longword to receive cache entry address
                     00C8   226 ;
                     00C8   227 ;   OUTPUTS:
                     00C8   228 ;
                     00C8   229 ;       r0 = True if symbol found
                     00C8   230 ;---
                     00C8   231
                     00C8   232 lookup_cache::
          3C   BB    00C8   233          pushr   #^m<r2, r3, r4, r5>
      FF70     30    00CA   234          bsbw    find_vbn_entry          ; look it up
  65    51   D0    00CD   235          movl    r1,(r5)                 ; return address
          3C   BA    00D0   236          popr    #^m<r2, r3, r4, r5>
               05    00D2   237          rsb
```

```
                     00D3    239          .sbttl  remove_cache, Remove a single cache entry
                     00D3    240 ;---
                     00D3    241 ;
                     00D3    242 ;       Remove a cache entry given the VBN of the entry.
                     00D3    243 ;
                     00D3    244 ; Inputs:
                     00D3    245 ;
                     00D3    246 ;       R0 = VBN of disk block
                     00D3    247 ;
                     00D3    248 ; Outputs:
                     00D3    249 ;
                     00D3    250 ;       r0 = status code
                     00D3    251 ;---
                     00D3    252
                     00D3    253 remove_cache::
     3C    BB      00D3    254          pushr   #^m<r2, r3, r4, r5>
   FF65    30      00D5    255          bsbw    find_vbn_entry          ; look it up
 0C 50     E9      00D8    256          blbc    r0,20$
     61    D0      00DB    257          movl    cache$l_link(r1),-
     62            00DD    258                  cache$l_link(r2)        ; unlink from the list
50   0E    D0      00DE    259          movl    #cache$c_length,r0      ; set length of block
   FF1C'   30      00E1    260          bsbw    dealloc_mem             ; deallocate cache entry
50   01    D0      00E4    261          movl    #1,r0
     3C    BA      00E7    262 20$:     popr    #^m<r2, r3, r4, r5>
           05      00E9    263          rsb
```

```
                                00EA   265          .sbttl  empty_cache, Empty disk block cache
                                00EA   266 ;---
                                00EA   267 ;
                                00EA   268 ;       Empty the entire disk block cache and return the
                                00EA   269 ;       storage used.
                                00EA   270 ;
                                00EA   271 ; Inputs:
                                00EA   272 ;
                                00EA   273 ;       None
                                00EA   274 ;
                                00EA   275 ; Outputs:
                                00EA   276 ;
                                00EA   277 ;       None
                                00EA   278 ;---
                                00EA   279
                        00FC    00EA   280          .entry  empty_cache,-
                                00EC   281                  ^m<r2,r3,r4,r5,r6,r7>
                                00EC   282
                    56    01    D0     00EC   283          movl    #1,r6                   ; preset success return
                55  0000'CF    9E     00EF   284          movab   dealloc_mem,r5          ; address of deallocation routine
                54  0000'CF    D0     00F4   285          movl    lbr$gl_control,r4       ; control table address
                  54    0E A4  D0     00F9   286          movl    lbr$l_ctxptr(r4),r4     ; context block address
                                00FD   287
                  50    4A A4  D0     00FD   288          movl    ctx$l_chdallsiz(r4),r0  ; cache header entry allocation block size
                  51    4E A4  D0     0101   289          movl    ctx$l_chdalladr(r4),r1  ; cache header entry allocation block addr
                              65 16     0105   290          jsb     (r5)                    ; deallocate the cache header entry blocks
                        4A A4  7C     0107   291          clrq    ctx$l_chdallsiz(r4)     ; clear the size and address in ctx block
                                010A   292
                        08 A4  9F     010A   293          pushab  ctx$l_cache(r4)         ; stack address of cache hash list pointer
                    54  08 A4  D0     010D   294          movl    ctx$l_cache(r4),r4      ; cache hash list address
                              9E  D4  0111   295          clrl    @(sp)+                  ; clear cache hash pointer in context block
                                0113   296
                53  0000007F 8F     D0     0113   297          movl    #<lbr$c_hashsize/4>-1,r3 ; start at end of table
                  52    6443  D0     011A   298 10$:     movl    (r4)[r3],r2             ; get entry for this hash bucket
                              34  13  011E   299          beql    30$                     ; if eql none
                    14 0C A2  E9     0120   300 20$:     blbc    cache$w_flags(r2),25$   ; branch if not modified
                  50    08 A2  D0     0124   301          movl    cache$l_address(r2),r0  ; address to write from
                  51    04 A2  D0     0128   302          movl    cache$l_vbn(r2),r1      ; and vbn to write at
                        09 56  E9     012C   303          blbc    r6,25$                  ; branch if previous write error
                         FECE' 30     012F   304          bsbw    write_block             ; write back the block
                        03 50  E8     0132   305          blbs    r0,25$                  ; branch if write error
                      56    50  D0     0135   306          movl    r0,r6                   ; save error for later
                50  00000200 8F     D0     0138   307 25$:     movl    #lbr$c_pagesize,r0      ; set size of block
                  51    08 A2  D0     013F   308          movl    cache$l_address(r2),r1  ; block address
                              65 16     0143   309          jsb     (r5)                    ; deallocate it
                      51    52  D0     0145   310          movl    r2,r1                   ; deallocate cache entry
                      50    0E  D0     0148   311          movl    #cache$c_length,r0
                      52    62  D0     014B   312          movl    cache$l_link(r2),r2     ; link to next
                              65 16     014E   313          jsb     (r5)                    ; deallocate cache entry
                              52 D5     0150   314          tstl    r2                      ; more in bucket?
                              CC 12     0152   315          bneq    20$                     ; if neq yes
                        C3 53  F4     0154   316 30$:     sobgeq  r3,10$                  ; loop for whole table
                      51    54  D0     0157   317          movl    r4,r1                   ; deallocate cache table
                50  00000200 8F     D0     015A   318          movl    #lbr$c_hashsize,r0
                              65 16     0161   319          jsb     (r5)
                      50    56  D0     0163   320          movl    r6,r0
                                04     0166   321          ret
```

```
0167   322
0167   323          .END
```

```
ADD_CACHE                       00000000 RG    02
ALLOC_CACHE_HDR                 00000078 R     02
CACHE$C_LENGTH                  0000000E
CACHE$K_LENGTH                  0000000E
CACHE$L_ADDRESS                 00000008
CACHE$L_LINK                    00000000
CACHE$L_VBN                     00000004
CACHE$W_FLAGS                   0000000C
CTX$B_DCTXTRFA                  00000022
CTX$B_EOMODRFA                  00000022
CTX$B_NXTPUTRFA                 0000003E
CTX$B_READRFA                   00000028
CTX$B_RPNEWTXT                  0000001C
CTX$C_CHDALLBLK               = 00000007
CTX$C_LENGTH                    00000086
CTX$K_LENGTH                    00000086
CTX$L_CACHE                     00000008
CTX$L_CHDALLADR                 0000004E
CTX$L_CHDALLSIZ                 0000004A
CTX$L_CTLFLG                    00000004
CTX$L_DCXCTX                    00000052
CTX$L_DCXMAPDSC                 00000056
CTX$L_DCXRECDSC                 0000005A
CTX$L_HIVBN                     00000046
CTX$L_RDBLKS                    0000003A
CTX$L_RDBUFR                    00000032
CTX$L_RDVBN1                    00000036
CTX$L_READBUF                   0000002E
CTX$L_RECRAB                    0000000C
CTX$L_RPHASHT                   00000010
CTX$L_RPLDESC                   00000014
CTX$W_IFI                       00000002
CTX$W_ISI                       00000000
DEALLOC_MEM                     ********   X   02
EMPTY_CACHE                     000000EA RG    02
FIND_VBN_ENTRY                  0000003D R     02
GET_ZMEM                        ********   X   02
LBR$C_HASHSIZE                = 00000200
LBR$C_PAGESIZE                = 00000200
LBR$GL_CONTROL                  ********   X   02
LBR$L_CTXPTR                  = 0000000E
LOOKUP_CACHE                    000000C8 RG    02
REMOVE_CACHE                    000000D3 RG    02
WRITE_BLOCK                     ********   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 |
| $ABS$ | 00000086 ( 134.) | 01 ( 1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| $CODE$ | 00000167 ( 359.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 29 | 00:00:00.09 | 00:00:00.73 |
| Command processing | 148 | 00:00:00.61 | 00:00:02.94 |
| Pass 1 | 172 | 00:00:03.14 | 00:00:06.99 |
| Symbol table sort | 0 | 00:00:00.22 | 00:00:00.43 |
| Pass 2 | 72 | 00:00:00.85 | 00:00:01.91 |
| Symbol table output | 7 | 00:00:00.05 | 00:00:00.05 |
| Psect synopsis output | 1 | 00:00:00.03 | 00:00:00.03 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 431 | 00:00:05.02 | 00:00:13.09 |

The working set limit was 1200 pages.
15285 bytes (30 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 196 non-local and 14 local symbols.
323 source lines were read in Pass 1, producing 16 object records in Pass 2.
11 pages of virtual memory were used to define 10 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[LBR.OBJ]LBR.MLB;1 | 2 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 5 |
| TOTALS (all libraries) | 7 |

271 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:CACHE/OBJ=OBJ$:CACHE MSRC$:CACHE/UPDATE=(ENH$:CACHE)+LIB$:LBR/LIB
```

CACHE
LIS

DATA
LIS

LBR

LBRSHR
MAP

OLDLIBFMT
MDL

DUMP
LIS

LBR
MDL

PREFIX
REQ

LBRUSR
SDL