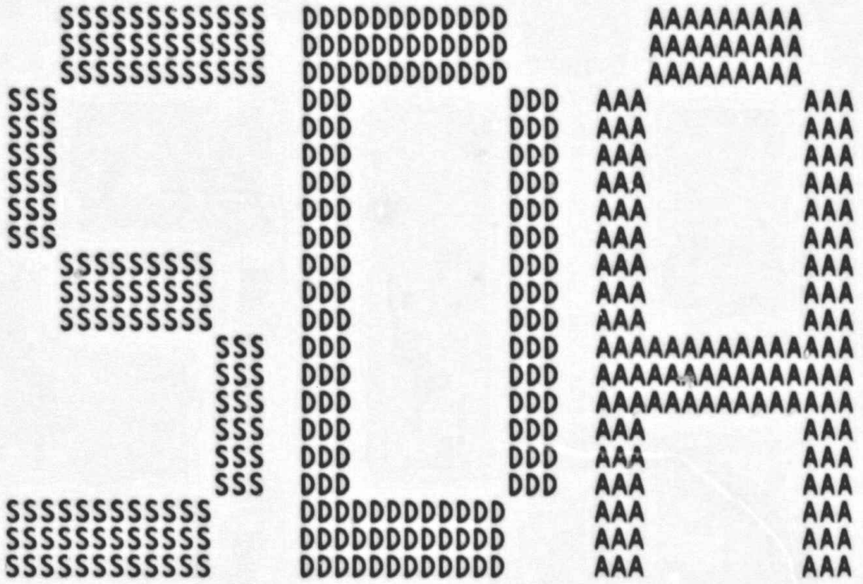
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA

```
PPPPPPPP      000000      000000    LL
PPPPPPPP      000000      000000    LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PP      PP  00      00  00      00  LL
PPPPPPPP    00      00  00      00  LL
PPPPPPPP    00      00  00      00  LL
PP          00      00  00      00  LL
PP          00      00  00      00  LL
PP          00      00  00      00  LL                ....
PP          00      00  00      00  LL                ....
PP            000000      000000    LLLLLLLLLL        ....
PP            000000      000000    LLLLLLLLLL        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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  POOL    DISPLAY NON-PAGED POOL ROUTINES
0000     2          .SBTTL  COPYRIGHT NOTICE
0000     3          .IDENT  'V04-000'
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     30          .SBTTL  PROGRAM DESCRIPTION
0000     31 ;++
0000     32 ;   FACILITY
0000     33 ;
0000     34 ;       SYSTEM DUMP ANALYZER
0000     35 ;
0000     36 ;   ABSTRACT
0000     37 ;
0000     38 ;       DUMP NON-PAGED POOL ROUTINES
0000     39 ;
0000     40 ;   ENVIRONMENT
0000     41 ;
0000     42 ;       NATIVE MODE, USER MODE
0000     43 ;
0000     44 ;   AUTHOR
0000     45 ;
0000     46 ;       TIM HALVORSEN, JULY 1978
0000     47 ;
0000     48 ;   MODIFIED BY
0000     49 ;
0000     50 ;       V03-013 MSH0069         Michael S. Harvey       23-Jul-1984
0000     51 ;               Clarify pool summary messages so that pool summaries
0000     52 ;               are useful on hardcopy terminals.
0000     53 ;
0000     54 ;       V03-012 MSH0063         Michael S. Harvey       16-Jul-1984
0000     55 ;               Correct bounds check once again to allow recognition of a KFE.
0000     56 ;
0000     57 ;       V03-011 TMK0001         Todd M. Katz            28-Apr-1984
0000     58 ;               Replace LOG type with LNM type in BLOCK_TABLE. Also, fix the
0000     59 ;               check that is made within the routine CHECK_BLOCK for an unknown
0000     60 ;               block's granularity. There are two problems with the current
0000     61 ;               default granularity check. First, the check is not being made
0000     62 ;               on the block's size. Secondly, the check itself is being made
0000     63 ;               incorrectly
0000     64 ;
0000     65 ;       V03-010 MSH0023         Michael S. Harvey       23-Mar-1984
0000     66 ;               Add KFPB to data structure recognition table.
0000     67 ;
0000     68 ;       V03-009 EMD0063         Ellen M. Dusseault      17-Mar-1984
0000     69 ;               Modify macro, $EQU, to create two tables for
0000     70 ;               identifying subtypes of the generic code.
0000     71 ;               Tables are used by the format command.
0000     72 ;
0000     73 ;       V03-008 MSH0014         Michael S. Harvey       2-Mar-1984
0000     74 ;               Correct granularity checks for PQBs.
0000     75 ;
0000     76 ;       V03-007 MSH0012         Michael S. Harvey       27-Feb-1984
0000     77 ;               Correct granularity checks for KFE and KFD data structures
0000     78 ;               so that SDA correctly identifies them in paged pool.
0000     79 ;
0000     80 ;       V03-006 MSH0007         Michael S. Harvey       3-Feb-1984
0000     81 ;               Process Quota Blocks are now in paged pool. Changed data
0000     82 ;               here so that they are recognised when seen.
0000     83 ;
0000     84 ;       V03-005 MSH0004         Michael S. Harvey       2-Feb-1984
0000     85 ;               Change upper bounds check for global section descriptors
0000     86 ;               to support longer section names. Also, correct lower bound
```

```
0000     87 ;                  check for PFN-mapped global section descriptors.
0000     88 ;
0000     89 ;       V03-004 CDS0001         Christian D. Saether     3-Aug-1983
0000     90 ;                Remove obsolete $RVXDEF.
0000     91 ;
0000     92 ;       V03-003 JLV0279         Jake VanNoy              27-JUL-1983
0000     93 ;                Remove obsolete $BRDDEF.
0000     94 ;
0000     95 ;       V03-002 RPG0002         Bob Grosso               27-Jun-1983
0000     96 ;                Replace old Known file structures with the new ones.
0000     97 ;
0000     98 ;       V03-001 KTA3062         Kerbey T. Altmann        26-Jun-1983
0000     99 ;                Account for extra sized FCB's.
0000    100 ;
0000    101 ;       V02-008 KTA0069         Kerbey T. Altmann        24-Jan-1982
0000    102 ;                Add SHOW POOL/SRP.
0000    103 ;
0000    104 ;       V02-007 KTA0062         Kerbey T. Altmann        04-Jan-1982
0000    105 ;                Modify a few global locations to access new pool
0000    106 ;                sizes and starting addresses.
0000    107 ;
0000    108 ;       V02-006 KTA0044         Kerbey T. Altmann        11-Nov-1981
0000    109 ;                Add SHOW POOL/LRP.
0000    110 ;
0000    111 ;       V03-005 KTA0029         Kerbey Altmann  01-Aug-1981
0000    112 ;                1. Add more entries to BLOCK_TABLE.
0000    113 ;                2. Add SHOW POOL range.
0000    114 ;                3. Add SHOW POOL/HEADER and /FREE
0000    115 ;
0000    116 ;       V03-004 KTA0027         Kerbey Altmann  28-Jul-1981
0000    117 ;                Modify to accept sub-typable blocks.
0000    118 ;
0000    119 ;       V003    MTR0001         Mike Rhodes      22-Jun-1981
0000    120 ;                Change all CMPW's referencing an MSG$_ symbol to CMPL's.
0000    121 ;                Change default addressing mode to longword.
0000    122 ;                Remove references to $SDAMSGDEF macro.
0000    123 ;
0000    124 ;       V002    TMH0002         Tim Halvorsen    07-Feb-1981
0000    125 ;                Convert word displacements to longword displacements
0000    126 ;                Fix end-of-pool edge condition when dumping a pool
0000    127 ;                which ends with a block in use.
0000    128 ;
0000    129 ;       V001    TMH0001         Tim Halvorsen    30-Sep-1980
0000    130 ;                Use $EQU rather than $EQULST to obtain block type
0000    131 ;                code strings because MDL no longer produces $EQULST.
0000    132 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ;       SYMBOL DEFINTIONS
                ;
                        .nocross
                        $OPTDEF                 ; OPTION DEFINITIONS
                        $ACBDEF
                        $ADPDEF                 ; DEFINE SYMBOLS FOR BLOCK TABLE
                        $AQBDEF
                        $CDDBDEF
                        $CDRPDEF
                        $CEBDEF
                        $CRBDEF
                        $CXBDEF
                        $DDBDEF
                        $DPTDEF
                        $FCBDEF
                        $FKBDEF
                        $GSDDEF
                        $IDBDEF
                        $IRPDEF
                        $IRPEDEF
                        $JIBDEF
                        $KFRHDEF
                        $KFEDEF
                        $KFDDEF
                        $KFPBDEF
                        $LNMSTRDEF
                        $MTLDEF
                        $MVLDEF
                        $PBHDEF
                        $PCBDEF
                        $PDBDEF
                        $PFLDEF
                        $PQBDEF
                        $PTRDEF
                        $RBMDEF
                        $RSBDEF
                        $RVTDEF
                        $SHBDEF
                        $TQEDEF
                        $UCBDEF
                        $VCADEF
                        $VCBDEF
                        $VECDEF
                        $WCBDEF
                        .cross

000000D0                IRP_SIZE =      <IRP$C_LENGTH+^XF>&<^C<^XF>>
```

```
                     0000   186          .SBTTL  STORAGE DEFINITIONS
                     0000   187
                     0000   188 ;
                     0000   189 ;       STORAGE DEFINITIONS
                     0000   190 ;
                     0000   191          .default displacement,long
                     0000   192
                     0000   193          .PSECT  SDADATA,NOEXE,WRT
                     0000   194
                     0000   195 ;
                     0000   196 ; NOTE: Following two locations must be contiguous!!!
                     0000   197 ;
                     0000   198 SPACE_USED:
00000004             0000   199          .BLKL   1                       ; TOTAL SPACE IN USE
                     0004   200 TOTAL_SPACE:
00000008             0004   201          .BLKL   1                       ; TOTAL SPACE AVAILABLE
                     0008   202 IRP_POOL_START:
0000000C             0008   203          .BLKL   1                       ; START OF IRP POOL
                     000C   204 IRP_BITMAP:
00000000             000C   205          .LONG   0                       ; ADDRESS OF IRP POOL BITMAP
                     0010   206
                     0010   207 ;
                     0010   208 ;       DEFINE SELECTION CONDITIONS FOR VARIOUS BLOCK TYPES
                     0010   209 ;
                     0010   210
00000001             0010   211          NONP =  1                       ; BIT FOR NON-PAGED POOL
00000002             0010   212          PAGD =  2                       ; BIT FOR PAGED POOL
                     0010   213
00000001             0010   214          IRP  =  1                       ; CODE FOR IRP LOOKASIDE LIST
00000002             0010   215          LRP  =  2                       ; CODE FOR LRP LOOKASIDE LIST
00000003             0010   216          SRP  =  3                       ; CODE FOR SRP LOOKASIDE LIST
                     0010   217
                     0010   218          .MACRO  BLOCK   TYPE,MIN=0,MAX=<^X7FFF>,POOL=NONPAGED,GRAN=<^XF>
                     0010   219          .BYTE   DYN$C_'TYPE
                     0010   220          .WORD   MIN
                     0010   221          .WORD   <MAX+^XF>&^C<^XF>
                     0010   222          .IF     IDN,<POOL>,<NONPAGED>
                     0010   223          .BYTE   NONP
                     0010   224          .IFF
                     0010   225          .IF     IDN,<POOL>,<PAGED>
                     0010   226          .BYTE   PAGD
                     0010   227          .IFF
                     0010   228          .IF     IDN,<POOL>,<ANYPOOL>
                     0010   229          .BYTE   NONP!PAGD
                     0010   230          .IFF
                     0010   231          .ERROR  ; ILLEGAL POOL SPECIFICATION
                     0010   232          .ENDC
                     0010   233          .ENDC
                     0010   234          .ENDC
                     0010   235          .BYTE   GRAN
                     0010   236          .ENDM
                     0010   237
                     0010   238 BLOCK_TABLE:
                     0010   239          BLOCK   ACB,ACB$L_KAST+4,IRP$C_LENGTH
00000007             0017   240 BLK_TBL_SIZ=.-BLOCK_TABLE
                     0017   241          BLOCK   ADP,ADP$C_MBAADPLEN,1536
                     001E   242          BLOCK   AQB,AQB$C_LENGTH,AQB$C_LENGTH,GRAN=0
```

```
                0025   243          BLOCK   CDRP,CDRP$C_LENGTH,CDRP$C_LENGTH+CDRP$C_BT_LEN+16
                002C   244          BLOCK   CEB,IRP$C_LENGTH,IRP$C_LENGTH
                0033   245          BLOCK   CRB,CRB$C_LENGTH,CRB$L_INTD2+VEC$C_LENGTH
                003A   246          BLOCK   CXB,CXB$C_LENGTH
                0041   247          BLOCK   DDB,DDB$C_LENGTH,DDB$C_LENGTH
                0048   248          BLOCK   DPT,DPT$C_LENGTH,GRAN=0
                004F   249          BLOCK   ERP,0,0                                 ; NOT IN LOCAL MEMORY
                0056   250          BLOCK   EXTGSD,GSD$C_EXTGSDLNG,GSD$C_EXTGSDLNG+43,PAGED,GRAN=0
                005D   251          BLOCK   FCB,FCB$C_LENGTH,FCB$S_FCBDEF
                0064   252          BLOCK   FRK,FKB$C_LENGTH,FKB$C_LENGTH
                006B   253          BLOCK   GSD,GSD$C_LENGTH,GSD$C_LENGTH+43,PAGED,GRAN=0
                0072   254          BLOCK   IDB,IDB$C_LENGTH,IDB$C_LENGTH+8*4       ; EXTRA 8 DUE TO ERROR IN IN
                0079   255          BLOCK   IRP,IRP$C_LENGTH,IRP$C_LENGTH
                0080   256          BLOCK   IRPE,IRPE$C_LENGTH,IRP$C_LENGTH
                0087   257          BLOCK   JIB,JIB$C_LENGTH,JIB$C_LENGTH
                008E   258          BLOCK   KFRH,KFRH$C_LENGTH,2060,PAGED,GRAN=0 ; 12 header+max 4 blk hdr
                0095   259          BLOCK   KFE,KFE$C_LENGTH,KFE$C_MAXLEN,PAGED,GRAN=0      ;KFE
                009C   260          BLOCK   KFD,KFD$C_LENGTH,KFD$C_LENGTH+252,PAGED,GRAN=0  ;KFD+max devspc
                00A3   261          BLOCK   KFPB,KFPB$C_LENGTH,KFPB$C_LENGTH,PAGED,GRAN=0
                00AA   262          BLOCK   LNM,LNMB$T_NAME+2+1,,PAGED      ; MIN = CONSTANT PART LNMB + 2 BYTES
                00B1   263          BLOCK   MBX,0,0                                 ; NOT IN LOCAL MEMORY
                00B8   264          BLOCK   MTL,MTL$C_LENGTH,MTL$C_LENGTH,PAGED
                00BF   265          BLOCK   MVL,MVL$C_FIXLEN
                00C6   266          BLOCK   PBH,PBH$C_LENGTH,PBH$C_LENGTH
                00CD   267          BLOCK   PCB,PCB$C_LENGTH,PCB$C_LENGTH,GRAN=0
                00D4   268          BLOCK   PDB,PDB$C_LENGTH,PDB$C_LENGTH
                00DB   269          BLOCK   PFL,PFL$C_LENGTH
                00E2   270          BLOCK   PIB,0,0                                 ; NOT STANDARD BLOCK FORMAT
                00E9   271          BLOCK   PTR,PTR$L_PTR0,,ANYPOOL
                00F0   272          BLOCK   PQB,PQB$C_LENGTH,PQB$C_LENGTH,PAGED,GRAN=0
                00F7   273          BLOCK   RBM,RBM$C_LENGTH,GRAN=0
                00FE   274          BLOCK   RSB,RSB$C_LENGTH,RSB$C_LENGTH+RSB$K_MAXLEN+1
                0105   275          BLOCK   RVT,RVT$C_LENGTH
                010C   276          BLOCK   SHB,SHB$C_LENGTH,SHB$C_LENGTH
                0113   277          BLOCK   SHMCEB,0,0                              ; NEVER IN LOCAL MEMORY
                011A   278          BLOCK   SHMGSD,0,0                              ; NEVER IN LOCAL MEMORY
                0121   279          BLOCK   SHRBUFIO,0,0                            ; NEVER IN LOCAL MEMORY
                0128   280          BLOCK   SLAVCEB,CEB$C_LENGTH+12,CEB$C_LENGTH+12
                012F   281          BLOCK   TQE,TQE$C_LENGTH,TQE$C_LENGTH
                0136   282          BLOCK   UCB,UCB$C_LENGTH,GRAN=0
                013D   283          BLOCK   VCB,VCB$C_LENGTH,VCB$C_LENGTH,GRAN=0
                0144   284          BLOCK   WCB,WCB$C_LENGTH
          00    014B   285          .BYTE   0                               ; END OF TABLE
                014C   286
                014C   287
                014C   288          .MACRO  $DEFINI NAME,P1,P2
                014C   289 LAST_VALUE =     0
                014C   290 LAST_VALUE_MAIN = 0
                014C   291 SYM =           0
                014C   292 LAST_SYM =      0
                014C   293 SUBTF =         0               ; sub type field
                014C   294 NSUBT =         0               ; number of subtype fields for a generic function
                014C   295 OFFSET =        0               ; offset for the generic function into 2nd table
                014C   296                                 ; ( dyn_subptr).
                014C   297
                014C   298 ; The first two psects contain tables for subtypes.  The first table pointed
                014C   299 ; to by symbol, dyn_mainptr, contains offsets into the second table for each
```

```
014C    300 ; generic code.  The second table pointed to by symbol, dyn_subptr, contains
014C    301 ; offsets into the table (dyn_tab) of ascii symbols for each subtype of the
014C    302 ; generic code.
014C    303 ;
014C    304         .PSECT  DYNMAINPTR,LONG
014C    305 DYN_MAINPTR::
014C    306         .PSECT  DYNSUBPTR, LONG
014C    307 DYN_SUBPTR::
014C    308         .PSECT  DYNMAP,LONG
014C    309 DYN_MAP::
014C    310         .PSECT  DYNPTR,LONG
014C    311 DYN_PTR::
014C    312         .PSECT  DYNTAB
014C    313 DYN_TAB::
014C    314         .ASCIC  /UNKNOWN/
014C    315 LASTPC=.                                    ; **** TEMP UNTIL BUG IN MACRO FIXED **
014C    316         .ENDM   $DEFINI
014C    317
014C    318         .MACRO  $EQU SYMBOL,VALUE
014C    319         .PSECT  DYNTAB
014C    320         .=LASTPC
014C    321         .IF EQ <LAST_VALUE_MAIN-VALUE>
014C    322             .=.-L-1
014C    323         .ENDC
014C    324         LAST_SYM = .-DYN_TAB
014C    325         S = %LOCATE(< >,SYMBOL)+1
014C    326         L = %LENGTH(SYMBOL)-S
014C    327         .ASCIC  /%EXTRACT(S,L,SYMBOL)/
014C    328         LASTPC=.                            ; **** TEMP UNTIL BUG IN MACRO FIXED **
014C    329         .IF NE <LAST_VALUE_MAIN-VALUE>
014C    330             DIFF = VALUE-LAST_VALUE-1
014C    331             .IF EQ DIFF
014C    332                 .IF NE SUBTF
014C    333                     NSUBT = NSUBT+1
014C    334                    .PSECT DYNSUBPTR
014C    335                    .WORD LAST_SYM
014C    336                    OFFSET = .-DYN_SUBPTR
014C    337                 .IFF
014C    338                     .PSECT DYNMAP
014C    339                     .BYTE  0
014C    340                     .PSECT DYNPTR
014C    341                     .WORD  SYM
014C    342                 .ENDC
014C    343             .ENDC
014C    344             .IF GT DIFF
014C    345                 .IF NE SUBTF
014C    346                     .PSECT DYNMAP
014C    347                     .BYTE  NSUBT
014C    348                     .PSECT DYNPTR
014C    349                     .WORD  SYM
014C    350                     SUBTF = 0
014C    351                     DIFF = VALUE-LAST_VALUE_MAIN-1
014C    352                 .IFF
014C    353                     .PSECT DYNMAP
014C    354                     .BYTE  0
014C    355                     .PSECT DYNPTR
014C    356                     .WORD  SYM
```

```
                014C   357                     .ENDC
                014C   358                         .REPT  DIFF
                014C   359                           .PSECT DYNMAP
                014C   360                           .BYTE  -1
                014C   361                           .PSECT DYNPTR
                014C   362                           .WORD  0
                014C   363                         .ENDR
                014C   364                 .ENDC
                014C   365                 .IF LT DIFF
                014C   366                     SUBTF = 1
                014C   367                     NSUBT = 1
                014C   368                     .PSECT DYNMAINPTR
                014C   369                     .WORD OFFSET/2
                014C   370                     .PSECT DYNSUBPTR
                014C   371                     .WORD LAST_SYM
                014C   372                     OFFSET = .-DYN_SUBPTR
                014C   373                 .ENDC
                014C   374                 LAST_VALUE = VALUE
                014C   375                 .IF EQ SUBTF
                014C   376                     LAST_VALUE_MAIN = VALUE
                014C   377                     SYM = LAST_SYM
                014C   378                 .ENDC
                014C   379             .ENDC
                014C   380             SYMBOL = VALUE
                014C   381             .ENDM   $EQU
                014C   382
                014C   383             .MACRO  $DEFEND NAME,P1,P2
                014C   384 END_SYM=LAST_VALUE_MAIN+1
                014C   385             .PSECT  DYNTAB
                014C   386             .BYTE   -1
                014C   387             $EQU    END     256
                014C   388             .ENDM   $DEFEND
                014C   389             .nocross
                014C   390             $DYNDEF
                0200   391             .cross
                0200   392
            00000000   393             .PSECT  DYNCNT,LONG
                0000   394 DYN_CNT:
            00000102   0000   395             .BLKW   END_SYM
            00000102   0102   396 DYN_CNT_SIZ=.-DYN_CNT
            00000000   397             .PSECT  DYNBYTES,LONG
                0000   398 DYN_BYTES:
            00000204   0000   399             .BLKL   END_SYM
            00000204   0204   400 DYN_BYT_SIZ=.-DYN_BYTES
            00000003   0204   401             .MDELETE $DEFINI,$EQU,$DEFEND
                0204   402
            00000000   403             .PSECT  POOL,EXE,NOWRT
                0000   404
                0000   405             .DEFAULT DISPLACEMENT,LONG
```

```
                                0000   408          .SBTTL  SHOW_POOL_RANGE -- DISPLAY DYNAMIC STORAGE POOLS
                                0000   409 ;---
                                0000   410 ;
                                0000   411 ;       DISPLAY AND FORMAT THE CONTENTS OF THE NON-PAGED AND PAGED
                                0000   412 ;       DYNAMIC STORAGE POOL OR ANY RANGE THEREOF.
                                0000   413 ;
                                0000   414 ; INPUTS:
                                0000   415 ;
                                0000   416 ;       OPTIONS = OPTIONS FLAGS (RANGE OR LENGTH BITS RELEVANT)
                                0000   417 ;       ESP     = END OF POOL
                                0000   418 ;                 (OR, IF LENGTH BIT SET)
                                0000   419 ;       ESP     = SIZE OF POOL
                                0000   420 ;       ESP+4   = START OF POOL
                                0000   421 ;
                                0000   422 ; OUTPUTS:
                                0000   423 ;
                                0000   424 ;       NONE
                                0000   425 ;---
                                0000   426
                        0C7C    0000   427          .ENTRY  SHOW_POOL_RANGE,^M<R2,R3,R4,R5,R6,R10,R11>
                                0002   428
56  00000000'EF   9E    0002   429          MOVAB   OPTIONS, R6             ; POINT TO OPTIONS WORD
        52  66   D0    0009   430          MOVL    (R6), R2
     18 52   01   E0    000C   431          BBS     #OPT$V_IRP, R2, 20$     ; /IRP NOT ALLOWED
        52   0C   D3    0010   432          BITL    #OPT$M_NONPAGED!OPT$M_PAGED, R2 ; IF NEITHER SPECIFIED...
             04   12    0013   433          BNEQ    10$                     ; THEN
     00 66   02   E2    0015   434          BBSS    #OPT$V_NONPAGED,(R6),10$        ; SET /NONPAGED
51  00000000'EF   D0    0019   435 10$:     MOVL    ESP, R1                 ; POINT TO EXPRESSION STACK
     07 52   03   E0    0020   436          BBS     #OPT$V_RANGE, R2, 30$   ; RANGE SPECIFIED
     11 52   04   E0    0024   437          BBS     #OPT$V_LENGTH, R2, 40$  ; LENGTH SPECIFIED
             50   D4    0028   438 20$:     CLRL    R0                      ; SYNTAX ERROR
             04   002A   439          RET
                  002B   440
55  04 A1   D0    002B   441 30$:     MOVL    4(R1),R5                ; R5 = LOWEST ADDRESS
54  61  55   C3    002F   442          SUBL3   R5,(R1),R4              ; R4 = SIZE
05 66   04   E2    0033   443          BBSS    #OPT$V_LENGTH,(R6),50$  ; SET A SINGLE BIT FOR RANGE
        03   11    0037   444          BRB     50$
                  0039   445
    54  61   7D    0039   446 40$:     MOVQ    (R1),R4                 ; R5 = LOWEST ADDRESS
                  003C   447
        0204   31    003C   448 50$:     BRW     SCAN_POOL               ; JOIN COMMON CODE
```

```
                                003F   451          .SBTTL  SHOW_POOL -- DISPLAY DYNAMIC STORAGE POOLS
                                003F   452 ;---
                                003F   453 ;
                                003F   454 ;       SHOW_POOL
                                003F   455 ;
                                003F   456 ;       DISPLAY AND FORMAT THE CONTENTS OF THE NON-PAGED AND PAGED
                                003F   457 ;       DYNAMIC STORAGE POOL.
                                003F   458 ;
                                003F   459 ;   INPUTS:
                                003F   460 ;
                                003F   461 ;       NONE
                                003F   462 ;
                                003F   463 ;   OUTPUTS:
                                003F   464 ;
                                003F   465 ;       NONE
                                003F   466 ;
                                003F   467 ;---
                                003F   468
                                003F   469          .ENABLE LSB
                          0FFC  003F   470          .ENTRY  SHOW_POOL,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                0041   471
                00000000'EF   D4  0041   472          CLRL    SPACE_USED                              ; INITIALIZE SPACE USAGE
          0102 8F   00   6E   00  2C  0047   473          MOVC5   #0,(SP),#0,#DYN_CNT_SIZ,DYN_CNT ; AND TYPE COUNTS
                00000000'EF       004E
          0204 8F   00   6E   00  2C  0053   474          MOVC5   #0,(SP),#0,#DYN_BYT_SIZ,DYN_BYTES ; AND BYTE COUNTS
                00000000'EF       005A
                5B  00000000'EF   9E  005F   475          MOVAB   BUFFER,R11                              ; R11 = GETMEM BUFFER
                53  00000000'EF   9E  0066   476          MOVAB   OPTIONS,R3                              ; PTR TO OPTIONS WORD
                63  0000022E 8F   D3  006D   477          BITL    #OPT$M_IRP!OPT$M_LRP!OPT$M_SRP!-
                                       0074   478              OPT$M_NONPAGED!OPT$M_PAGED,(R3); SOMETHING SET?
                                 07    12  0074   479          BNEQ    10$                             ; BRANCH IF SO
                63   0000022E 8F   C8  0076   480          BISL    #OPT$M_IRP!OPT$M_LRP!OPT$M_SRP!-
                                       007D   481              OPT$M_NONPAGED!OPT$M_PAGED,(R3); SET /ALL
                70 63    01   E1  007D   482 10$:     BBC     #OPT$V_IRP,(R3),30$             ; SKIP IF NO /IRP
                                       0081   483 ;
                                       0081   484 ;       SCAN CONTENTS OF IRP LOOKASIDE LIST
                                       0081   485 ;
                                       0081   486          SUBHD   <IRP lookaside list>
                                       008E   487          SKIP    PAGE
                16 63    06   E0  0095   488          BBS     #OPT$V_SUMMARY,(R3),20$ ; SKIP IF SUMMARY ONLY
                                       0099   489          PRINT   0,<!_!_!_Dump of blocks allocated from IRP lookaside list>
                                       00A6   490          SKIP    3
                5A    D0 8F   9A  00AF   491 20$:     MOVZBL  #IRP_SIZE,R10           ; R10 = IRP BLOCK SIZE
                                       00B3   492          REQMEM  @EXE$GL_SPLITADR,R7     ; ADDRESS OF IRP LOOKASIDE LIST
                                       00C3   493          REQMEM  @IOC$GL_IRPCNT,R8       ; NUMBER OF IRP SLOTS ALLOCATED
                                       00D3   494          REQMEM  @IOC$GL_IRPFL,R9        ; HEAD OF IRP LIST
                56   00000000'EF   D0  00E3   495          MOVL    IOC$GL_IRPFL,R6         ; ADDRESS OF HEAD
                                 01    DD  00EA   496          PUSHL   #IRP                    ; INDICATE WHICH LOOKASIDE LIST
                          0506'CF   01  FB  00EC   497          CALLS   #1,W^DO_ILRP            ; SCAN THE IRP FREELIST
          03 00000000'EF   05   E0  00F1   498 30$:     BBS     #OPT$V_LRP,OPTIONS,35$  ; /LRP PRESENT?
                        009E   31   00F9   499          BRW     42$                     ; NO, SKIP THE LRP
                                       00FC   500 ;
                                       00FC   501 ;       SCAN CONTENTS OF LRP LOOKASIDE LIST
                                       00FC   502 ;
                00000000'EF   D4  00FC   503 35$:     CLRL    SPACE_USED                              ; INITIALIZE SPACE USAGE
          0102 8F   00   6E   00  2C  0102   504          MOVC5   #0,(SP),#0,#DYN_CNT_SIZ,DYN_CNT ; AND TYPE COUNTS
                00000000'EF       0109
```

```
                0204 8F    00    6E    00    2C  010E   505          MOVC5   #0,(SP),#0,#DYN_BYT_SIZ,DYN_BYTES ; AND BYTE COUNTS
                             00000000'EF        0115
                                                011A   506          SUBHD   <LRP lookaside list>
                                                0127   507          SKIP    PAGE
                   16 00000000'EF    06    E0  012E   508          BBS     #OPT$V_SUMMARY,OPTIONS,40$      ; SKIP IF SUMMARY ONLY
                                                0136   509          PRINT   0,<!_!_!_Dump of blocks allocated from LRP lookaside list>
                                                0143   510          SKIP    3
                                                014C   511 40$:     REQMEM  @IOC$GL_LRPSIZE,R10     ; R10 = LRP BLOCK SIZE
                                                015C   512          REQMEM  @IOC$GL_LRPSPLIT,R7     ; ADDRESS OF LRP LOOKASIDE LIST
                                                016C   513          REQMEM  @IOC$GL_LRPCNT,R8       ; NUMBER OF LRP SLOTS ALLOCATED
                                                017C   514          REQMEM  @IOC$GL_LRPFL,R9        ; HEAD OF IRP LIST
                      56   00000000'EF    D0  018C   515          MOVL    IOC$GL_LRPFL,R6         ; ADDRESS OF HEAD
                                   02    DD  0193   516          PUSHL   #LRP                    ; INDICATE WHICH LOOKASIDE LIST
                            0506'CF    01    FB  0195   517          CALLS   #1,W^DO_ILRP            ; SCAN THE LRP FREELIST
                   03 00000000'EF    09    E0  019A   518 42$:     BBS     #OPT$V_SRP,OPTIONS,45$  ; /SRP PRESENT?
                                009E   31    01A2   519          BRW     SCAN_POOL               ; NO, SKIP THE SRP
                                                01A5   520 ;
                                                01A5   521 ;       SCAN CONTENTS OF SRP LOOKASIDE LIST
                                                01A5   522 ;
                             00000000'EF    D4  01A5   523 45$:     CLRL    SPACE_USED                      ; INITIALIZE SPACE USAGE
                0102 8F    00    6E    00    2C  01AB   524          MOVC5   #0,(SP),#0,#DYN_CNT_SIZ,DYN_CNT ; AND TYPE COUNTS
                             00000000'EF        01B2
                0204 8F    00    6E    00    2C  01B7   525          MOVC5   #0,(SP),#0,#DYN_BYT_SIZ,DYN_BYTES ; AND BYTE COUNTS
                             00000000'EF        01BE
                                                01C3   526          SUBHD   <SRP lookaside list>
                                                01D0   527          SKIP    PAGE
                   16 00000000'EF    06    E0  01D7   528          BBS     #OPT$V_SUMMARY,OPTIONS,47$      ; SKIP IF SUMMARY ONLY
                                                01DF   529          PRINT   0,<!_!_!_Dump of blocks allocated from SRP lookaside list>
                                                01EC   530          SKIP    3
                                                01F5   531 47$:     REQMEM  @IOC$GL_SRPSIZE,R10     ; R10 = SRP BLOCK SIZE
                                                0205   532          REQMEM  @IOC$GL_SRPSPLIT,R7     ; ADDRESS OF SRP LOOKASIDE LIST
                                                0215   533          REQMEM  @IOC$GL_SRPCNT,R8       ; NUMBER OF SRP SLOTS ALLOCATED
                                                0225   534          REQMEM  @IOC$GL_SRPFL,R9        ; HEAD OF IRP LIST
                      56   00000000'EF    D0  0235   535          MOVL    IOC$GL_SRPFL,R6         ; ADDRESS OF HEAD
                                   03    DD  023C   536          PUSHL   #SRP                    ; INDICATE WHICH LOOKASIDE LIST
                            0506'CF    01    FB  023E   537          CALLS   #1,W^DO_ILRP            ; SCAN THE SRP FREELIST
                                                0243   538 ;
                                                0243   539 ;       SCAN CONTENTS OF DYNAMIC NON-PAGED POOL
                                                0243   540 ;
                                                0243   541 SCAN_POOL:
                      5A   00000000'EF    D0  0243   542          MOVL    OPTIONS,R10             ; PICK UP OPTIONS
                             60 5A    02    E1  024A   543          BBC     #OPT$V_NONPAGED,R10,60$ ; BRANCH IF NOT SPECIFIED
                                                024E   544          SUBHD   <Non-paged dynamic storage pool>
                                                025B   545          SKIP    PAGE
                              16 5A    06    E0  0262   546          BBS     #OPT$V_SUMMARY,R10,50$ ; SKIP IF SUMMARY ONLY
                                                0266   547          PRINT   0,<!_!_!_Dump of blocks allocated from non-paged pool>
                                                0273   548          SKIP    3
                                                027C   549 50$:
                56   04    00000000'EF    C1  027C   550          ADDL3   EXE$GL_NONPAGED,#4,R6   ; ADDRESS OF FREE SPACE LIST
                                                0284   551          REQMEM  @MMG$GL_NPAGEDYN,R2     ; NO, USE BEGINNING OF NON-PAGED POOL
                                                0294   552          REQMEM  @MMG$GL_NPAGNEXT        ; ENDING ADDRESS OF POOL
                         51    01FF  8F    AA  02A1   553          BICW    #^X1FF,R1               ; LOW 9 BITS ARE FLAGS
                               51    52    C2  02A6   554          SUBL2   R2,R1                   ; LENGTH OF STORAGE POOL
                               53    01    D0  02A9   555          MOVL    #NONP,R3                ; BITS FOR NON-PAGED
                                     5F    10  02AC   556          BSBB    SET_UP_DUMP             ; GO DUMP IT
                                                02AE   557 ;
                                                02AE   558 ;       SCAN CONTENTS OF DYNAMIC PAGED POOL
```

```
                                02AE    559 ;
                     57 5A  03  E1  02AE    560 60$:    BBC     #OPT$V_PAGED,R10,80$     ; BRANCH IF NOT SPECIFIED
                                02B2    561         SUBHD   <Paged dynamic storage pool>
                                02BF    562         SKIP    PAGE
                     16 5A  06  E0  02C6    563         BBS     #OPT$V_SUMMARY,R10,70$  ; SKIP IF SUMMARY ONLY
                                02CA    564         PRINT   0,<!_!_!_Dump of blocks allocated from paged pool>
                                02D7    565         SKIP    3
                                02E0    566 70$:
                56   00000000'EF  D0  02E0    567         MOVL    EXE$GL_PAGED,R6          ; ADDRESS OF FREE SPACE LIST
                                02E7    568         REQMEM  @MMG$GL_PAGEDYN,R2       ; BEGINNING OF PAGED POOL
                                02F7    569         REQMEM  @SGN$GL_PAGEDYN          ; NO, USE LENGTH OF POOL
                        53  02  D0  0304    570         MOVL    #PAGD,R3                 ; BITS FOR PAGED
                            04  10  0307    571         BSBB    SET_UP_DUMP              ; DUMP IT
                                0309    572 80$:
                        50  01  D0  0309    573         MOVL    #1,R0                    ; SUCCESS
                            04  030C    574         RET
                                030D    575         .DISABLE LSB
                                030D    576
                                030D    577 ;
                                030D    578 ;       R1 = MAXIMUM LENGTH
                                030D    579 ;       R2 = MINIMUM START ADDRESS
                                030D    580 ;       R3 = POOL TYPE BITS
                                030D    581 ;       R4 = LENGTH (IF RANGE)
                                030D    582 ;       R5 = START ADDRESS (IF RANGE)
                                030D    583 ;       R6 = HEAD OF POOL
                                030D    584 ;
                                030D    585 SET_UP_DUMP:
                     1C 5A  04  E1  030D    586         BBC     #OPT$V_LENGTH,R10,30$    ; RANGE SET?
                        52  55  D1  0311    587         CMPL    R5,R2                    ; ADDRESS MUST BE START OR GREATER
                            03  1E  0314    588         BGEQU   10$                      ; OKAY
                        55  52  D0  0316    589         MOVL    R2,R5                    ; IT'S NOT, USE MIN
                50      52  51  C1  0319    590 10$:    ADDL3   R1,R2,R0                 ; MAX ENDING ADDRESS
                7E      50  55  C3  031D    591         SUBL3   R5,R0,-(SP)              ; GET MAX LENGTH
                        6E  54  D1  0321    592         CMPL    R4,(SP)                  ; USE GIVEN MUST BE LESS OR EQUAL
                            03  1B  0324    593         BLEQU   20$                      ; OKAY
                        54  6E  D0  0326    594         MOVL    (SP),R4                  ; IT'S NOT, USE MAX
                            8E  D5  0329    595 20$:    TSTL    (SP)+                    ; CLEAN STACK
                            03  11  032B    596         BRB     40$                      ; JOIN COMMON
                                032D    597
                        54  51  7D  032D    598 30$:    MOVQ    R1,R4                    ; USE THE DEFAULT
                      0078  8F  BB  0330    599 40$:    PUSHR   #^M<R3,R4,R5,R6>         ; PUSH THE PARAMETERS
              39'AF     04  FB  0334    600         CALLS   #4,B^DUMP_POOL           ; DUMP THE POOL
                                0338    601         RSB
```

```
                                0339   604          .SBTTL  DUMP_POOL, DISPLAY DYNAMIC STORAGE POOL
                                0339   605 ;---
                                0339   606 ;
                                0339   607 ;       THIS ROUTINE DISPLAYS THE CONTENTS (AS BEST IT CAN) OF
                                0339   608 ;       A GIVEN DYNAMIC STORAGE POOL.
                                0339   609 ;
                                0339   610 ; INPUTS:
                                0339   611 ;       4(AP)  = BIT MASK FOR TYPE OF POOL
                                0339   612 ;       8(AP)  = LENGTH OF POOL
                                0339   613 ;       12(AP) = ADDRESS OF BEGINNING OF STORAGE POOL
                                0339   614 ;       16(AP) = ADDRESS OF FREE STORAGE LISTHEAD
                                0339   615 ;
                                0339   616 ;---
                                0339   617          .ENABLE LSB
                                0339   618
                                0339   619 DUMP_POOL:
                         08FC   0339   620          .WORD   ^M<R2,R3,R4,R5,R6,R7,R11>
                                033B   621
        5B   00000000'EF   9E   033B   622          MOVAB   L^BUFFER,R11            ; R11 = SCRATCH BUFFER
                     04 AB   D4   0342   623          CLRL    4(R11)                  ; ZERO LENGTH
                00000000'EF   D4   0345   624          CLRL    SPACE_USED              ; INITIALIZE SPACE USAGE
  0102 8F    00    6E    00   2C   034B   625          MOVC5   #0,(SP),#0,#DYN_CNT_SIZ,DYN_CNT ; ZERO COUNTS
                00000000'EF          0352
  0204 8F    00    6E    00   2C   0357   626          MOVC5   #0,(SP),#0,#DYN_BYT_SIZ,DYN_BYTES ; ZERO BYTE COUNTS
                00000000'EF          035E
             52    0C AC   D0   0363   627          MOVL    12(AP),R2               ; SET ADDRESS OF POOL
          54    08 AC   52   C1   0367   628          ADDL3   R2,8(AP),R4             ; ENDING ADDRESS OF POOL
    00000004'EF    08 AC   D0   036C   629          MOVL    8(AP),TOTAL_SPACE       ; TOTAL SPACE IN POOL
                           0374   630          REQMEM  @16(AP),R3              ; END+1 OF FIRST USED AREA
                           0381   631
             53   52   D1   0381   632          CMPL    R2,R3                   ; IS FIRST BLOCK FREE?
                  38   1B   0384   633          BLEQU   50$                     ; BRANCH IF NOT
             56   53   D0   0386   634 10$:     MOVL    R3,R6                   ; START OF FREE AREA
                           0389   635 ;
                           0389   636 ; WE ARE NOW POINTING TO THE NEXT FREE CHUNK.  USING ITS LINK POINTER
                           0389   637 ; AND ITS LENGTH, COMPUTE THE STARTING AND ENDING ADDRESS OF THE NEXT
                           0389   638 ; CHUNK OF ALLOCATED MEMORY.
                           0389   639 ;
                           0389   640 20$:     REQMEM  (R6),(R11),#8           ; LINK ADDRESS + LENGTH
       52   04 AB   56   C1   0396   641          ADDL3   R6,4(R11),R2            ; ADDRESS FOLLOWING FREE BLOCK
             57   52   D0   039B   642          MOVL    R2,R7                   ; MAKE A COPY
          0C AC   52   D1   039E   643          CMPL    R2,12(AP)               ; IS IT LESS THAN START?
                  04   1E   03A2   644          BGEQU   30$                     ; NO, OKAY
             52    0C AC   D0   03A4   645          MOVL    12(AP),R2               ; USE SPECIFIED START ADDRESS
             53   6B   D0   03A8   646 30$:     MOVL    (R11),R3                ; ADDRESS FOLLOWING USED AREA
                  03   12   03AB   647          BNEQ    40$                     ; BRANCH IF NOT END OF FREE LIST
             53   54   D0   03AD   648          MOVL    R4,R3                   ; IF END OF FREE LIST, USE END OF POOL
          0C AC   53   D1   03B0   649 40$:     CMPL    R3,12(AP)               ; WITHIN RANGE OF SPECIFIED ADDRESS?
                  D0   1F   03B4   650          BLSSU   10$                     ; NO, LOOK FOR NEXT BLOCK
             54   57   D1   03B6   651          CMPL    R7,R4                   ; CHECK IF END OF POOL
                  03   1F   03B9   652          BLSSU   50$                     ; BRANCH IF NOT YET DONE
                00B7   31   03BB   653          BRW     120$
                           03BE   654
  11 00000000'EF   00   E1   03BE   655 50$:     BBC     #OPT$V_FREE,OPTIONS,55$ ; IF /FREE THEN
          57   04 AB   D0   03C6   656          MOVL    4(R11),R7               ; LENGTH OF FREE AREA
                  0B   13   03CA   657          BEQL    55$                     ; NONE
             7E   01   CE   03CC   658          MNEGL   #1,-(SP)                ; SET FOR "[free]"
```

```
                7E    56    7D  03CF     659          MOVQ    R6,-(SP)                        ; LENGTH AND ADDRESS
      0623'     CF    03    FB  03D2     660          CALLS   #3,W^POOL_BLOCK                 ; PRINT IT OUT
                                03D7     661 ;
                                03D7     662 ; DETERMINE THE BOUNDS OF THE BLOCK WE ARE NOW POINTING TO.
                                03D7     663 ;
                                03D7     664 55$:     REQMEM  (R2),(R11),#12                  ; MUST BE AT LEAST 3 LONGWORDS
          55    08    AB    3C  03E4     665          MOVZWL  IRP$W_SIZE(R11),R5              ; GET LENGTH OF BLOCK
                      15    13  03E8     666          BEQL    60$                             ; BRANCH IF BAD LENGTH
          51    53    52    C3  03EA     667          SUBL3   R2,R3,R1                        ; MAXIMUM ALLOWABLE LENGTH
                      51    55  D1  03EE  668          CMPL    R5,R1                           ; CHECK FOR REASONABLE LENGTH
                      0C    14  03F1     669          BGTR    60$                             ; BRANCH IF BAD LENGTH
                      55    0F  CO  03F3 670          ADDL2   #^XF,R5                         ; ROUND TO NEAREST 16 BYTES
                      55    0F  CA  03F6 671          BICL2   #^XF,R5
                         00AF     30  03F9 672          BSBW    CHECK_BLOCK                     ; MAKE ADDITIONAL BLOCK CHECKS
                        33 50     E8  03FC 673          BLBS    R0,90$                          ; BRANCH IF VALID TYPE
                                03FF     674 ;
                                03FF     675 ; IF WE CANNOT MAKE SENSE OUT OF THE BLOCK CONTENTS, LOOKAHEAD AND
                                03FF     676 ; TRY TO LOCATE THE NEXT BLOCK WHICH LOOKS REASONABLE.
                                03FF     677 ;
                      56    52  D0  03FF 678 60$:     MOVL    R2,R6                           ; INITIALIZE LOOKAHEAD POINTER
                      56    10  CO  0402 679 70$:     ADDL2   #16,R6                          ; LOOKAHEAD EACH 16 BYTES
                      53    56  D1  0405 680          CMPL    R6,R3                           ; CHECK IF END OF AREA
                            22    1E  0408 681          BGEQU   80$                             ; BRANCH IF END
                                040A     682          REQMEM  (R6),(R11),#12                  ; FIRST 3 LONGWORDS
          51    08    AB    3C  0417     683          MOVZWL  IRP$W_SIZE(R11),R1              ; CHECK IF SIZE VALID
                      E5    13  041B     684          BEQL    70$                             ; SKIP IF BAD LENGTH
          50    53    52    C3  041D     685          SUBL3   R2,R3,R0                        ; MAXIMUM ALLOWABLE LENGTH
                      50    51  D1  0421  686          CMPL    R1,R0                           ; CHECK FOR REASONABLE LENGTH
                      DC    1A  0424     687          BGTRU   70$                             ; SKIP IF BAD LENGTH
                         0082     30  0426 688          BSBW    CHECK_BLOCK                     ; MAKE ADDITIONAL BLOCK CHECKS
                        D6 50     E9  0429 689          BLBC    R0,70$                          ; BRANCH IF ILLEGAL BLOCK
                                042C     690 80$:
          55    56    52    C3  042C     691          SUBL3   R2,R6,R5                        ; LENGTH OF BLOCK
                      51    D4  0430     692          CLRL    R1                              ; USE 0 AS BLOCK TYPE
                                0432     693 ;
                                0432     694 ; THE BOUNDS OF THE BLOCK HAVE BEEN DETERMINED - DISPLAY THE CONTENTS
                                0432     695 ;
      00000000'EF41             B6  0432 696 90$:     INCW    DYN_CNT[R1]                     ; INCREMENT COUNT FOR TYPE
00000000'EF41       55    CO  0439     697          ADDL    R5,DYN_BYTES[R1]                ; INCREMENT BYTES USED FOR THIS TYPE
                      51    DD  0441     698          PUSHL   R1                              ; TYPE OF BLOCK
                      05    13  0443     699          BEQL    100$                            ; SKIP SUBTYPE IF ZERO
          01 AE     0B    AB  88  0445 700          BISB    IRP$B_TYPE+1(R11),1(SP)         ; SET POSSIBLE SUBTYPE
                      55    DD  044A     701 100$:     PUSHL   R5                              ; LENGTH OF BLOCK
                      52    DD  044C     702          PUSHL   R2                              ; ADDRESS OF BLOCK
                0623'CF     03  FB  044E 703          CALLS   #3,W^POOL_BLOCK                 ; DUMP POOL BLOCK
                      52    55  CO  0453 704          ADDL    R5,R2                           ; INCREMENT POINTER
          00000000'EF       55  CO  0456 705          ADDL    R5,SPACE_USED                   ; INCREMENT SPACE USAGE
                      53    52  D1  045D 706          CMPL    R2,R3                           ; CHECK IF END OF USED AREA
                      08    1E  0460     707          BGEQU   110$                            ; BRANCH IF END
                      54    52  D1  0462 708          CMPL    R2,R4                           ; CHECK IF END OF RANGE
                      0E    1E  0465     709          BGEQU   120$                            ; BRANCH IF END
                         FF6D     31  0467 710          BRW     55$                             ; CONTINUE IF NOT
                                046A     711 ;
                                046A     712 ; END OF USED AREA REACHED - SKIP TO NEXT FREE CHUNK AND LOOP
                                046A     713 ;
                      54    53  D1  046A 714 110$:     CMPL    R3,R4                           ; ARE WE ARE THE END OF THE POOL?
                      06    1E  046D     715          BGEQU   120$                            ; BRANCH IF SO
```

```
                56    53   D0  046F   716          MOVL    R3,R6                   ; SKIP TO NEXT FREE BLOCK
                   FF14   31  0472   717          BRW     20$
                          0475   718 ;
                          0475   719 ; END OF POOL REACHED - DISPLAY STATISTICS
                          0475   720 ;
                          0475   721 120$:   SKIP    PAGE
              02    04 AC   D1  047C   722          CMPL    4(AP),#PAGD             ; PAGED POOL BEING DISPLAYED?
                       0F   13  0480   723          BEQL    125$                    ; IF EQL YES
                          0482   724          PRINT   0,<Summary of non-paged pool contents>
                       0D   11  048F   725          BRB     126$
                          0491   726 125$:   PRINT   0,<Summary of paged pool contents>
          06E1'CF    00   FB  049E   727 126$:   CALLS   #0,W^SHOW_COUNTS          ; DISPLAY TYPE COUNTS
                          04A3   728 130$:
                          04A3   729          STATUS  SUCCESS
                       04  04AA   730          RET
                          04AB   731
                          04AB   732          .DISABLE        LSB
```

```
                                04AB   735 ;
                                04AB   736 ;       LOCAL SUBROUTINE TO CHECK IF BLOCK IS VALID
                                04AB   737 ;
                                04AB   738 ; INPUTS:
                                04AB   739 ;       R7    = SCRATCH REGISTER
                                04AB   740 ;       R11   = ADDRESS OF FIRST 16 BYTES IN LOCAL STORAGE
                                04AB   741 ;       4(AP) = MASK FOR TYPE OF POOL
                                04AB   742 ;
                                04AB   743 ; OUTPUTS:
                                04AB   744 ;       R0    = STATUS
                                04AB   745 ;       R1    = BLOCK TYPE
                                04AB   746 ;
                                04AB   747
                                04AB   748 CHECK_BLOCK:
                                04AB   749 ;
                                04AB   750 ; CHECK THE TYPE - 1) IS IT NON-ZERO, 2) IS IT A TYPE DEFINED IN THE
                                04AB   751 ; $DYNDEF MACRO, 2) IF IT IS SUBTYPEABLE, IS THE SUBTYPE WITHIN RANGE?
                                04AB   752 ;
          51    0A AB    9A   04AB   753          MOVZBL  IRP$B_TYPE(R11),R1      ; PICK UP TYPE
                   52    13   04AF   754          BEQL    60$                     ; BRANCH IF NOT VALID TYPE
    57  00000000'EF41    90   04B1   755          MOVB    DYN_MAP[R1],R7          ; CHECK VALIDITY
                   48    19   04B9   756          BLSS    60$                     ; BRANCH IF NOT VALID TYPE
                   0B    13   04BB   757          BEQL    10$                     ; BRANCH IF NOT SUBTYPABLE
          50    0B AB    90   04BD   758          MOVB    IRP$B_TYPE+1(R11),R0    ; PICK UP POSSIBLE SUBTYPE
                   05    13   04C1   759          BEQL    10$                     ; ZERO IS OKAY
             57    50    91   04C3   760          CMPB    R0,R7                   ; CHECK RANGE
                   3B    1A   04C6   761          BGTRU   60$                     ; OUT OF RANGE, NOT VALID
                                04C8   762 ;
                                04C8   763 ; NOW MAKE SPECIAL CHECKS ON PARAMETERS DEFINED IN BLOCK_TABLE -
                                04C8   764 ; MAX AND MIN SIZE, TYPE OF POOL, AND GRANULARITY.
                                04C8   765 ;
50  00000010'EF    9E   04C8   766 10$:    MOVAB   BLOCK_TABLE,R0          ; ADDRESS START OF TABLE
                   60    95   04CF   767 20$:    TSTB    (R0)                    ; CHECK IF END OF TABLE
                   27    13   04D1   768          BEQL    50$                     ; BRANCH IF TABLE EXAUSTED
             80    51    91   04D3   769          CMPB    R1,(R0)+                ; CHECK IF TYPE IN TABLE
                   05    13   04D6   770          BEQL    30$                     ; BRANCH IF FOUND
             50    06    C0   04D8   771          ADDL    #BLK_TBL_SIZ-1,R0       ; SKIP TO NEXT ENTRY IN TABLE
                   F2    11   04DB   772          BRB     20$                     ; AND LOOP UNTIL DONE
                                04DD   773
          57    08 AB    3C   04DD   774 30$:    MOVZWL  IRP$W_SIZE(R11),R7      ; PICK UP SIZE
             80    57    B1   04E1   775          CMPW    R7,(R0)+                ; CHECK AGAINST MINIMUM
                   1D    1F   04E4   776          BLSSU   60$                     ; BRANCH IF ILLEGAL
             80    57    B1   04E6   777          CMPW    R7,(R0)+                ; CHECK AGAINST MAXIMUM
                   18    1A   04E9   778          BGTRU   60$                     ; BRANCH IF ILLEGAL
          80    04 AC    93   04EB   779          BITB    4(AP),(R0)+             ; CAN THIS BLOCK BE IN THIS POOL?
                   12    13   04EF   780          BEQL    60$                     ; NO, LEAVE
             80    57    93   04F1   781          BITB    R7,(R0)+                ; CHECK GRANULARITY OF BLOCK
                   0D    12   04F4   782          BNEQ    60$                     ; BRANCH IF NOT GRANULAR
             50    01    D0   04F6   783 40$:    MOVL    #1,R0                   ; MARK BLOCK VALID
                         05   04F9   784          RSB
                                04FA   785
          57    08 AB    3C   04FA   786 50$:    MOVZWL  IRP$W_SIZE(R11),R7      ; PICK UP SIZE AND TEST GRANULARITY
             0F    57    93   04FE   787          BITB    R7,#^X0F                ; DEFAULT GRANULARITY = 16 BYTES
                   F3    13   0501   788          BEQL    40$                     ; BRANCH IF GRANULARITY OK
                                0503   789
                   50    7C   0503   790 60$:    CLRQ    R0                      ; MARK BLOCK ILLEGAL - UNKNOWN
                         05   0505   791          RSB
```

```
                                0506   794 ;
                                0506   795 ;       LOCAL SUBROUTINE TO CREATE LRP-IRP USAGE BITMAP
                                0506   796 ;
                                0506   797 ;       R6  =   Lookaside list head
                                0506   798 ;       R7 -->  Start of pool for lookaside list
                                0506   799 ;       R8  =   Number of blocks in lookaside list
                                0506   800 ;       R9 -->  Lookaside list head
                                0506   801 ;       R10 =   Size of block
                                0506   802 ;       R11-->  Scratch buffer
                                0506   803 ;
                                0506   804
                                0506   805         .ENABLE LSB
                                0506   806 DO_ILRP:
                          0000  0506   807         .WORD   0
                                0508   808
                     58   DD   0508   809         PUSHL   R8                      ; HOLD
          50    58    FD 8F   78   050A   810         ASHL    #-3,R8,R0               ; CALC LENGTH OF BITMAP
             7E    50   01   C1   050F   811         ADDL3   #1,R0,-(SP)             ; NUMBER OF BYTES FOR BITMAP
          58    0000000C'EF   D0   0513   812         MOVL    IRP_BITMAP,R8           ; ADDRESS OF BITMAP
                     13   12   051A   813         BNEQ    5$                      ; BRANCH IF ALREADY ALLOCATED
                     6E   DD   051C   814         PUSHL   (SP)                    ; AMOUNT NEEDED
          00000000'EF   01   FB   051E   815         CALLS   #1,ALLOCATE             ; ALLOCATE THE SPACE FOR THE BITMAP
          0000000C'EF   51   D0   0525   816         MOVL    R1,IRP_BITMAP           ; AND SAVE ADDRESS OF IT
                     58   51   D0   052C   817         MOVL    R1,R8
    68    6E    00   6E   00   2C   052F   818 5$:     MOVC5   #0,(SP),#0,(SP),(R8)    ; ZERO BITMAP
                     54   8E   7D   0535   819         MOVQ    (SP)+,R4                ; CLEAN STACK & R5 = # OF BLOCKS
                     56   59   D1   0538   820 10$:    CMPL    R9,R6                   ; CHECK IF END OF LIST
                     20   13   053B   821         BEQL    30$                     ; BRANCH IF END OF LIST
                50   59   57   C3   053D   822         SUBL3   R7,R9,R0                ; CALCULATE OFFSET FROM START OF IRPS
                     0C   15   0541   823         BLEQ    20$                     ; BRANCH IF ILLEGAL POOL ADDRESS
                     50   5A   C6   0543   824         DIVL    R10,R0                  ; GET INDEX INTO BITMAP
                     55   50   D1   0546   825         CMPL    R0,R5                   ; CHECK IF BIT NUMBER TOO LARGE
                     04   1E   0549   826         BGEQU   20$                     ; BRANCH IF OUTSIDE OF BITMAP
             00 68   50   E2   054B   827         BBSS    R0,(R8),20$             ; SET BIT IN BITMAP FOR THIS FREE IRP
                          054F   828 20$:    REQMEM  (R9),R9                 ; GET FIRST LONGWORD AND FOLLOW CHAIN
                     DB   11   055B   829         BRB     10$
                          055D   830
00000004'EF    55   5A   C5   055D   831 30$:    MULL3   R10,R5,TOTAL_SPACE      ; TOTAL BYTES FOR IRP LIST
53    00000004'EF   57   C1   0565   832         ADDL3   R7,TOTAL_SPACE,R3       ; ENDING ADDRESS OF SCAN
                     56   D4   056D   833         CLRL    R6                      ; INIT BITMAP INDEX
                          056F   834 ;
                          056F   835 ;       SCAN THE IRP LOOKASIDE POOL FOR ALLOCATED BLOCKS
                          056F   836 ;
                     53   57   D1   056F   837 40$:    CMPL    R7,R3                   ; CHECK IF DONE WITH SCAN
                          57   18   0572   838         BGEQ    100$                    ; BRANCH IF DONE
          03 0000000C'FF   56   E1   0574   839         BBC     R6,@IRP_BITMAP,45$      ; BRANCH IF BLOCK NOT ON FREE LIST
                     0090   31   057C   840         BRW     110$                    ; BLOCK ON FREE LIST
             00000000'EF   5A   C0   057F   841 45$:    ADDL    R10,SPACE_USED          ; INCREMENT SPACE USAGE
                          0586   842         REQMEM  (R7),(R11),#32          ; GET ENOUGH TO USE
                50   0A AB   9A   0593   843         MOVZBL  IRP$B_TYPE(R11),R0      ; GET BLOCK TYPE CODE
                     09   13   0597   844         BEQL    50$                     ; BRANCH IF UNKNOWN
                00000000'EF40   95   0599   845         TSTB    DYN_MAP[R0]             ; CHECK IF TYPE LEGAL
                     02   18   05A0   846         BGEQ    60$                     ; BRANCH IF OK
                     50   D4   05A2   847 50$:    CLRL    R0                      ; ZERO = UNKNOWN
                00000000'EF40   B6   05A4   848 60$:    INCW    DYN_CNT[R0]             ; INCREMENT COUNT FOR TYPE
             00000000'EF40   5A   C0   05AB   849         ADDL    R10,DYN_BYTES[R0]       ; INCREMENT BYTES USED FOR TYPE
                     50   DD   05B3   850 70$:    PUSHL   R0                      ; BLOCK TYPE
```

```
                05   13  05B5   851            BEQL    80$                             ; SKIP SUBTYPE IF ZERO
      01 AE     0B   AB  88 05B7   852            BISB    IRP$B_TYPE+1(R11),1(SP)         ; SET POSSIBLE SUBTYPE
                5A   DD  05BC   853 80$:          PUSHL   R10                             ; BLOCK LENGTH
                57   DD  05BE   854            PUSHL   R7                              ; BLOCK ADDRESS
         23'AF      03   FB  05C0   855            CALLS   #3,B^POOL_BLOCK                 ; DUMP CONTENTS OF BLOCK
             57     5A   C0  05C4   856 90$:      ADDL    R10,R7                          ; NEXT BLOCK
                56   D6  05C7   857            INCL    R6                              ; INCREMENT BITMAP INDEX
                A4   11  05C9   858            BRB     40$                             ; CONTINUE UNTIL DONE
                         05CB   859
                         05CB   860 100$:      SKIP    PAGE
          04 AC      03   D1  05D2   861            CMPL    #SRP,4(AP)                      ; IS THIS THE SRP LIST?
                24   13  05D6   862            BEQL    108$                            ; IF EQL YES
          04 AC      02   D1  05D8   863            CMPL    #LRP,4(AP)                      ; IS THIS THE LRP LIST?
                0F   13  05DC   864            BEQL    107$                            ; IF EQL YES
                         05DE   865            PRINT   0,<Summary of IRP lookaside list>
                1C   11  05EB   866            BRB     109$
                         05ED   867 107$:      PRINT   0,<Summary of LRP lookaside list>
                0D   11  05FA   868            BRB     109$
                         05FC   869 108$:      PRINT   0,<Summary of SRP lookaside list>
       06E1'CF      00   FB  0609   870 109$:      CALLS   #0,W^SHOW_COUNTS                ; DISPLAY BLOCK TYPE COUNTS
                04   060E   871            RET
                         060F   872
AD 00000000'EF      00   E1  060F   873 110$:      BBC     #OPT$V_FREE,OPTIONS,90$ ; SKIP BLOCK UNLESS /FREE
             7E     01   CE  0617   874            MNEGL   #1,-(SP)                        ; SET TO PRINT "[Free]"
                A0   11  061A   875            BRB     80$                             ; DO IT
                         061C   876            .DISABLE LSB
```

```
                                061C   879 ;
                                061C   880 ;       LOCAL SUBROUTINE TO DUMP CONTENTS OF POOL BLOCK
                                061C   881 ;
                                061C   882 ;       4(AP) = ADDRESS OF BLOCK
                                061C   883 ;       8(AP) = LENGTH OF BLOCK
                                061C   884 ;       12(AP) = TYPE OF BLOCK (0 IF UNKNOWN)
                                061C   885 ;       13(AP) = POSSIBLE SUBTYPE
                                061C   886 ;
                                061C   887
                                061C   888         .ENABL  LSB
                                061C   889
                                061C   890 FREE_STR:
    5D 65 65 72 46 5B 00'       061C   891         .ASCIC  /[Free]/
                          06    061C
                                0623   892
                                0623   893 POOL_BLOCK:
                          007C  0623   894         .WORD   ^M<R2,R3,R4,R5,R6>
                                0625   895
    56   00000000'EF    D0    0625   896         MOVL    OPTIONS,R6              ; PICK UP THE OPTIONS WORD
         01 56    06    E1    062C   897         BBC     #OPT$V_SUMMARY,R6,5$    ; SKIP IF SUMMARY ONLY
                          04    0630   898         RET
                                0631   899
               0C AC      D5    0631   900 5$:     TSTL    12(AP)                  ; FLAG SET?
                    06    18    0634   901         BGEQ    15$                     ; NORMAL
          54   E3 AF      DE    0636   902         MOVAL   FREE_STR,R4             ; ADDRESS OF "[FREE]"
                    4F    11    063A   903         BRB     25$                     ; PRINT IT
                          063C   904
          50   0C AC      9A    063C   905 15$:     MOVZBL  12(AP),R0               ; TYPE OF BLOCK
    54   00000000'EF40    3C    0640   906         MOVZWL  DYN_PTR[R0],R4          ; OFFSET FROM DYN_TAB TO SYMBOL NAME
    54   00000000'EF44    9E    0648   907         MOVAB   DYN_TAB[R4],R4          ; ADDRESS OF SYMBOL NAME
    51   00000000'EF40    9A    0650   908         MOVZBL  DYN_MAP[R0],R1          ; PICK UP INFO ON TYPE
                    14    13    0658   909         BEQL    20$                     ; NOT SUB-TYPABLE, CONTINUE
          52   0D AC      9A    065A   910         MOVZBL  13(AP),R2               ; PICK UP POSSIBLE SUBTYPE
                    0E    13    065E   911         BEQL    20$                     ; NONE, CONT
               51   52    91    0660   912         CMPB    R2,R1                   ; CHECK LEGAL RANGE
                    09    14    0663   913         BGTR    20$                     ; OUT OF RANGE, CONT
               50   84    9A    0665   914 10$:     MOVZBL  (R4)+,R0                ; LENGTH OF SYMBOL
               54   50    C0    0668   915         ADDL    R0,R4                   ; STEP OVER IT
                 F7 52    F5    066B   916         SOBGTR  R2,10$                  ; DO IT UNTIL SYMBOL VALUE
          19 56     08    E1    066E   917 20$:     BBC     #OPT$V_TYPE,R6,25$      ; SKIP IF NO /TYPE=
               55   54    D0    0672   918         MOVL    R4,R5                   ; COPY POINTER
               50   85    9A    0675   919         MOVZBL  (R5)+,R0                ; PICK UP LENGTH
    00000000'EF     50    91    0678   920         CMPB    R0,STRUCTURE            ; DO LENGTHS MATCH?
                    5F    12    067F   921         BNEQ    90$                     ; NO, SO NO HOPE OF CHAR MATCH
00000004'FF    65   50    29    0681   922         CMPC3   R0,(R5),@STRUCTURE+4    ; DO CHAR MATCH?
               55   12    0689   923         BNEQ    90$                     ; NO, NO MATCH AT ALL
          08 AC     DD    068B   924 25$:     PUSHL   8(AP)                   ; LENGTH OF BLOCK
          04 AC     DD    068E   925         PUSHL   4(AP)                   ; ADDRESS OF BLOCK
               54   DD    0691   926         PUSHL   R4                      ; PUSH ADDRESS OF SYMBOL
                          0693   927         PRINT   3,<!7AC !8XL !5UL>
                          06A0   928 30$:
                          06A0   929         REQMEM  @4(AP),BUFFER,#16       ; 16 BYTES PER LINE
    50   00000000'EF     DE    06B2   930         MOVAL   BUFFER,R0
                    50    DD    06B9   931         PUSHL   R0                      ; ADDRESS OF DATA
                    10    DD    06BB   932         PUSHL   #16                     ; LENGTH OF STRING
                    80    DD    06BD   933         PUSHL   (R0)+
                    80    DD    06BF   934         PUSHL   (R0)+
```

```
                80   DD  06C1   935          PUSHL   (R0)+
                80   DD  06C3   936          PUSHL   (R0)+
                     06C5   937          PRINT   6,<!3( )!4(9XL) !AF>    ; PRINT DUMP LINE
          04 AC 10   C0  06D2   938          ADDL2   #16,4(AP)               ; INCREMENT ADDRESS
          08 AC 10   C2  06D6   939          SUBL2   #16,8(AP)               ; DECREMENT LENGTH
                04   15  06DA   940          BLEQ    90$                     ; DONE, LEAVE
          C0 56 07   E1  06DC   941          BBC     #OPT$V_HEADER,R6,30$    ; MORE TO DO
                     06E0   942 90$:
                04   06E0   943          RET
                     06E1   944
                     06E1   945          .DSABL  LSB
```

```
                                06E1   948 ;
                                06E1   949 ;       LOCAL SUBROUTINE TO DISPLAY THE BLOCK TYPE COUNTS
                                06E1   950 ;
                                06E1   951
                                06E1   952         .ENABL  LSB
                                06E1   953
                                06E1   954 SHOW_COUNTS:
                         03FC   06E1   955         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                                06E3   956
                                06E3   957         SKIP    1                       ; PRINT BLANK LINE
      52   00000000'EF    9E   06EC   958         MOVAB   DYN_PTR,R2              ; ADDRESS OF SYMBOL TABLE
      53   00000000'EF    9E   06F3   959         MOVAB   DYN_CNT,R3              ; ADDRESS OF COUNT TABLE
      54   00000000'EF    9E   06FA   960         MOVAB   DYN_BYTES,R4            ; ADDRESS OF BYTE USAGE TABLE
         57    0081  8F    3C   0701   961         MOVZWL  #END_SYM,R7             ; NUMBER OF SYMBOLS TO LOOK AT
      58   00000000'EF    7D   0706   962         MOVQ    SPACE_USED,R8           ; PICK UP SOME VALUES
   08 00000000'EF    04    E1   070D   963         BBC     #OPT$V_LENGTH,OPTIONS,5$; FOR RANGE SPECIFIED
            59    58    D1   0715   964         CMPL    R8,R9                   ; IS SPACE_USED > TOTAL_SPACE?
                  03    1B   0718   965         BLEQU   5$                      ; NO
            59    58    D0   071A   966         MOVL    R8,R9                   ; YES, THEN SET EQUAL
55  58    00000064  8F    C7   071D   967 5$:     DIVL3   #100,R8,R5              ; COMPUTE SPACE USED/100
                              0725   968 10$:
            51    84    D0   0725   969         MOVL    (R4)+,R1                ; GET # BYTES FOR THIS TYPE
            56    82    3C   0728   970         MOVZWL  (R2)+,R6                ; OFFSET FOR SYMBOL
            50    83    3C   072B   971         MOVZWL  (R3)+,R0                ; GET NEXT COUNT
                  1C    13   072E   972         BEQL    20$                     ; SKIP IF ZERO
         7E   51    55    C7   0730   973         DIVL3   R5,R1,-(SP)             ; GET PERCENTAGE USAGE
                  51    DD   0734   974         PUSHL   R1                      ; GET BYTE COUNT FOR THIS TYPE
      00000000'EF46    9F   0736   975         PUSHAB  DYN_TAB[R6]             ; ADDRESS OF SYMBOL
                  50    DD   073D   976         PUSHL   R0
                              073F   977         PRINT   4,<!5UW  !9AC = !8UL  (!UL%)>
                              074C   978 20$:
            D6 57    F4   074C   979         SOBGEQ  R7,10$
                              074F   980
         7E   59    58    C3   074F   981         SUBL3   R8,R9,-(SP)             ; SPACE LEFT TO USE
              7E    58    7D   0753   982         MOVQ    R8,-(SP)                ; TOTAL SPACE AVAILABLE
                              0756   983         SKIP    1
                              075F   984         PRINT   3,<Total space used = !UL out of !UL total bytes, !UL bytes left>
58    00000064  8F    C4   076C   985         MULL    #100,R8                 ; SET UP FOR PERCENTAGE
   7E   58    59    C7   0773   986         DIVL3   R9,R8,-(SP)
                              0777   987         SKIP    1
                              0780   988         PRINT   1,<Total space utilization = !UL%>
                        04    078D   989         RET
                              078E   990
                              078E   991         .DSABL  LSB
```

```
078E   993
078E   994          .END
```

```
ACB$L_KAST                      =  00000018
ADP$C_MBAADPLEN                 =  00000030
ALLOCATE                           ********    X   0A
AQB$C_LENGTH                    =  0000001C
ARGS                            =  00000003
BLK_TBL_SIZ                     =  00000007
BLOCK_TABLE                        00000010 R       02
BUFFER                             ********    X   0A
CDRP$C_BT_LEN                   =  00000040
CDRP$C_LENGTH                   =  0000002C
CEB$C_LENGTH                    =  00000038
CHECK_BLOCK                        000004AB R       0A
CRB$C_LENGTH                    =  00000048
CRB$L_INTD2                     =  00000048
CXB$C_LENGTH                    =  00000028
DDB$C_LENGTH                    =  00000044
DIFF                            =  0000007F
DO_ILRP                            00000506 R       0A
DPT$C_LENGTH                    =  00000038
DUMP_POOL                          00000339 R       0A
DYN$C_ACB                       =  00000002
DYN$C_ACL                       =  0000003F
DYN$C_ADP                       =  00000001
DYN$C_AQB                       =  00000003
DYN$C_BOOTCB                    =  00000006
DYN$C_BRDCST                    =  0000001A
DYN$C_BUFIO                     =  00000013
DYN$C_CDB                       =  00000033
DYN$C_CDRP                      =  00000039
DYN$C_CD_BBRPG                  =  00000002
DYN$C_CD_CDDB                   =  00000001
DYN$C_CD_SHDW_WRK               =  00000003
DYN$C_CEB                       =  00000004
DYN$C_CHIP                      =  00000048
DYN$C_CI                        =  00000061
DYN$C_CIA                       =  00000045
DYN$C_CIDG                      =  0000003B
DYN$C_CIMSG                     =  0000003C
DYN$C_CI_BDT                    =  00000001
DYN$C_CI_FQDT                   =  00000002
DYN$C_CLASSDRV                  =  00000064
DYN$C_CLU                       =  00000065
DYN$C_CLU_BTX                   =  00000004
DYN$C_CLU_CLUB                  =  00000003
DYN$C_CLU_CLUDCB                =  00000005
DYN$C_CLU_CLUOPT                =  00000006
DYN$C_CLU_CLUVEC                =  00000002
DYN$C_CLU_CSB                   =  00000001
DYN$C_CLU_LCKDIR                =  00000007
DYN$C_CONF                      =  00000007
DYN$C_CRB                       =  00000005
DYN$C_CST                       =  00000008
DYN$C_CXB                       =  0000001B
DYN$C_DCCB                      =  00000027
DYN$C_DDB                       =  00000006
DYN$C_DPT                       =  0000001E
DYN$C_ERP                       =  0000003A
DYN$C_EXTGSD                    =  00000028
DYN$C_FCB                       =  00000007
DYN$C_FRK                       =  00000008
DYN$C_GSD                       =  00000015
DYN$C_IDB                       =  00000009
DYN$C_INIT                      =  00000063
DYN$C_IRP                       =  0000000A
DYN$C_IRPE                      =  0000002C
DYN$C_JIB                       =  0000002F
DYN$C_JNL                       =  00000067
DYN$C_JNLWCB                    =  00000024
DYN$C_JNL_ABL                   =  00000001
DYN$C_JNL_ACBM                  =  00000004
DYN$C_JNL_ADL                   =  00000002
DYN$C_JNL_BCB                   =  00000003
DYN$C_JNL_BUF                   =  00000005
DYN$C_JNL_BXSTS                 =  00000013
DYN$C_JNL_CWQ                   =  00000010
DYN$C_JNL_DB                    =  00000006
DYN$C_JNL_DIOREAD               =  00000015
DYN$C_JNL_JMT                   =  00000009
DYN$C_JNL_MSG                   =  00000012
DYN$C_JNL_MSGDATA               =  00000014
DYN$C_JNL_NDL                   =  00000008
DYN$C_JNL_RC                    =  00000011
DYN$C_JNL_RCPC                  =  0000000C
DYN$C_JNL_RM                    =  0000000A
DYN$C_JNL_RRP                   =  0000000B
DYN$C_JNL_RUL                   =  0000000D
DYN$C_JNL_SFT                   =  00000007
DYN$C_JNL_VCL                   =  0000000E
DYN$C_JNL_VLE                   =  0000000F
DYN$C_JPB                       =  0000001F
DYN$C_KFD                       =  00000043
DYN$C_KFE                       =  00000018
DYN$C_KFPB                      =  00000044
DYN$C_KFRH                      =  00000026
DYN$C_LC_CHREML                 =  00000006
DYN$C_LC_CLS                    =  00000005
DYN$C_LC_FPEMUL                 =  00000007
DYN$C_LC_MP                     =  00000003
DYN$C_LC_MSCP                   =  00000008
DYN$C_LC_SCS                    =  00000004
DYN$C_LC_SYSL                   =  00000009
DYN$C_LKB                       =  00000035
DYN$C_LKID                      =  00000037
DYN$C_LNM                       =  00000040
DYN$C_LOADCODE                  =  00000062
DYN$C_LOG                       =  0000000B
DYN$C_LPD                       =  00000034
DYN$C_MBX                       =  0000002B
DYN$C_MPWMAP                    =  00000004
DYN$C_MTL                       =  00000019
DYN$C_MVL                       =  00000016
DYN$C_NDB                       =  0000001C
DYN$C_NET                       =  00000017
DYN$C_NON_PAGED                 =  00000001
```

```
DYN$C_ORB                       =  00000049
DYN$C_PAGED                     =  00000002
DYN$C_PBH                       =  00000020
DYN$C_PCB                       =  0000000C
DYN$C_PCBVEC                    =  00000001
DYN$C_PDB                       =  00000021
DYN$C_PFB                       =  00000047
DYN$C_PFL                       =  00000023
DYN$C_PGD                       =  00000066
DYN$C_PGD_F11BC                 =  00000001
DYN$C_PHVEC                     =  00000002
DYN$C_PIB                       =  00000022
DYN$C_PMB                       =  00000046
DYN$C_PQB                       =  0000000D
DYN$C_PRCMAP                    =  00000005
DYN$C_PTR                       =  00000025
DYN$C_RBM                       =  00000031
DYN$C_RIGHTSLIST                =  00000042
DYN$C_RSB                       =  00000036
DYN$C_RSHT                      =  00000038
DYN$C_RVT                       =  0000000E
DYN$C_SCS                       =  00000060
DYN$C_SCS_CDL                   =  00000001
DYN$C_SCS_CDT                   =  00000002
DYN$C_SCS_DIR                   =  00000003
DYN$C_SCS_HQB                   =  0000000B
DYN$C_SCS_PB                    =  00000004
DYN$C_SCS_PDT                   =  00000005
DYN$C_SCS_RDT                   =  00000006
DYN$C_SCS_SB                    =  00000007
DYN$C_SCS_SPNB                  =  00000009
DYN$C_SCS_SPPB                  =  00000008
DYN$C_SCS_UQB                   =  0000000A
DYN$C_SHB                       =  0000002A
DYN$C_SHMCEB                    =  0000002E
DYN$C_SHMGSD                    =  00000029
DYN$C_SHRBUFIO                  =  00000080
DYN$C_SLAVCEB                   =  0000002D
DYN$C_SPECIAL                   =  00000080
DYN$C_SSB                       =  0000001D
DYN$C_SUBTYPE                   =  00000060
DYN$C_SWPMAP                    =  00000003
DYN$C_TQE                       =  0000000F
DYN$C_TWP                       =  00000030
DYN$C_TYPAHD                    =  00000014
DYN$C_UCB                       =  00000010
DYN$C_UNUSED_2                  =  00000041
DYN$C_VCA                       =  00000032
DYN$C_VCB                       =  00000011
DYN$C_WCB                       =  00000012
DYN$C_WQE                       =  0000003E
DYN$C_XWB                       =  0000003D
DYN_BYTES                          00000000 R      09
DYN_BYT_SIZ                     =  00000204
DYN_CNT                            00000000 R      08
DYN_CNT_SIZ                     =  00000102
DYN_MAINPTR                        00000000 RG     03
DYN_MAP                            00000000 RG     05
DYN_PTR                            00000000 RG     06
DYN_SUBPTR                         00000000 RG     04
DYN_TAB                            00000000 RG     07
END                             =  00000100
END_SYM                         =  00000081
ESP                                ********    X   0A
EXE$GL_NONPAGED                    ********    X   0A
EXE$GL_PAGED                       ********    X   0A
EXE$GL_SPLITADR                    ********    X   0A
FCB$C_LENGTH                    =  000000B4
FCB$S_FCBDEF                    =  000000B4
FKB$C_LENGTH                    =  00000018
FREE_STR                           0000061C R      0A
GSD$C_EXTGSDLNG                 =  00000031
GSD$C_LENGTH                    =  00000023
IDB$C_LENGTH                    =  00000038
IOC$GL_IRPCNT                      ********    X   0A
IOC$GL_IRPFL                       ********    X   0A
IOC$GL_LRPCNT                      ********    X   0A
IOC$GL_LRPFL                       ********    X   0A
IOC$GL_LRPSIZE                     ********    X   0A
IOC$GL_LRPSPLIT                    ********    X   0A
IOC$GL_SRPCNT                      ********    X   0A
IOC$GL_SRPFL                       ********    X   0A
IOC$GL_SRPSIZE                     ********    X   0A
IOC$GL_SRPSPLIT                    ********    X   0A
IRP                             =  00000001
IRP$B_TYPE                      =  0000000A
IRP$C_LENGTH                    =  000000C4
IRP$W_SIZE                      =  00000008
IRPE$C_LENGTH                   =  00000058
IRP_BITMAP                         0000000C R      02
IRP_POOL_START                     00000008 R      02
IRP_SIZE                        =  000000D0
JIB$C_LENGTH                    =  00000074
KFD$C_LENGTH                    =  00000011
KFE$C_LENGTH                    =  00000037
KFE$C_MAXLEN                    =  0000005E
KFPB$C_LENGTH                   =  00000010
KFRH$C_LENGTH                   =  0000000C
L                               =  FFFFFFFF
LASTPC                          =  0000038D R      07
LAST_SYM                        =  0000038C
LAST_VALUE                      =  00000100
LAST_VALUE_MAIN                 =  00000100
LNMB$T_NAME                     =  00000011
LRP                             =  00000002
MMG$GL_NPAGEDYN                    ********    X   0A
MMG$GL_NPAGNEXT                    ********    X   0A
MMG$GL_PAGEDYN                     ********    X   0A
MSG$_SUCCESS                       ********    X   0A
MTL$C_LENGTH                    =  00000018
MVL$C_FIXLEN                    =  00000024
NEW_PAGE                           ********    X   0A
NONP                            =  00000001
NSUBT                           =  00000015
```

```
OFFSET                          =  0000007C
OPT$M_IRP                       =  00000002
OPT$M_LRP                       =  00000020
OPT$M_NONPAGED                  =  00000004
OPT$M_PAGED                     =  00000008
OPT$M_SRP                       =  00000200
OPT$V_FREE                      =  00000000
OPT$V_HEADER                    =  00000007
OPT$V_IRP                       =  00000001
OPT$V_LENGTH                    =  00000004
OPT$V_LRP                       =  00000005
OPT$V_NONPAGED                  =  00000002
OPT$V_PAGED                     =  00000003
OPT$V_RANGE                     =  00000003
OPT$V_SRP                       =  00000009
OPT$V_SUMMARY                   =  00000006
OPT$V_TYPE                      =  00000008
OPTIONS                            ********    X   0A
PAGD                            =  00000002
PBH$C_LENGTH                    =  00000200
PCB$C_LENGTH                    =  00000120
PDB$C_LENGTH                    =  00000034
PFL$C_LENGTH                    =  00000024
POOL_BLOCK                         00000623 R      0A
PQB$C_LENGTH                    =  000008C8
PRINT                              ********    X   0A
PTR$L_PTR0                      =  00000010
RBM$C_LENGTH                    =  0000000C
REQMEM                             ********    X   0A
RSB$C_LENGTH                    =  00000050
RSB$K_MAXLEN                    =  0000001F
RVT$C_LENGTH                    =  00000044
S                               =  00000004
SCAN_POOL                          00000243 R      0A
SET_READING                        ********    X   0A
SET_UP_DUMP                        0000030D R      0A
SGN$GL_PAGEDYN                     ********    X   0A
SHB$C_LENGTH                    =  00000020
SHOW_COUNTS                        000006E1 R      0A
SHOW_POOL                          0000003F RG     0A
SHOW_POOL_RANGE                    00000000 RG     0A
SKIP_LINES                         ********    X   0A
SPACE_USED                         00000000 R      02
SRP                             =  00000003
STRUCTURE                          ********    X   0A
SUBTF                           =  00000000
SYM                             =  0000038C
TOTAL_SPACE                        00000004 R      02
TQE$C_LENGTH                    =  00000030
UCB$C_LENGTH                    =  00000090
VCB$C_LENGTH                    =  000000EC
VEC$C_LENGTH                    =  00000024
WCB$C_LENGTH                    =  00000030
```

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
SDADATA	0000014C (332.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
DYNMAINPTR	00000010 (16.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
DYNSUBPTR	0000007C (124.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
DYNMAP	00000100 (256.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
DYNPTR	00000200 (512.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
DYNTAB	0000038D (909.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DYNCNT	00000102 (258.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
DYNBYTES	00000204 (516.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
POOL	0000078E (1934.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
LITERALS	00000339 (825.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.06	00:00:00.93
Command processing	115	00:00:00.42	00:00:03.21
Pass 1	805	00:00:29.32	00:01:40.75
Symbol table sort	0	00:00:02.49	00:00:10.97
Pass 2	200	00:00:05.89	00:00:22.03
Symbol table output	34	00:00:00.18	00:00:00.3[illegible]
Psect synopsis output	3	00:00:00.04	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1193	00:00:38.40	00:02:18.29

The working set limit was 2250 pages.
247978 bytes (485 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2413 non-local and 89 local symbols.
994 source lines were read in Pass 1, producing 66 object records in Pass 2.
59 pages of virtual memory were used to define 52 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SDA.OBJ]SDALIB.MLB;1	7
_255DUA28:[SYS.OBJ]LIB.MLB;1	41
_255DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	51

2619 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

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

PROCESS
LIS

RMS
LIS

QAST
LIS

RELEASE
LIS

POOL
LIS