```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM       MM      11        000000    PPPPPPPP    EEEEEEEEEE  NN       NN
RRRRRRRR    MM       MM      11        000000    PPPPPPPP    EEEEEEEEEE  NN       NN
RR      RR  MMMM   MMMM    1111      00      00  PP      PP  EE          NN       NN
RR      RR  MMMM   MMMM    1111      00      00  PP      PP  EE          NN       NN
RR      RR  MM  MM  MM      11        00      00  PP      PP  EE          NNNN     NN
RR      RR  MM  MM  MM      11        00      00  PP      PP  EE          NNNN     NN
RRRRRRRR    MM      MM      11        00      00  PPPPPPPP    EEEEEEEE    NN  NN   NN
RRRRRRRR    MM      MM      11        00      00  PPPPPPPP    EEEEEEEE    NN  NN   NN
RR  RR      MM      MM      11        00      00  PP          EE          NN    NNNN
RR  RR      MM      MM      11        00      00  PP          EE          NN    NNNN
RR    RR    MM      MM      11        00      00  PP          EE          NN      NN      ....
RR    RR    MM      MM      11        00      00  PP          EE          NN      NN      ....
RR      RR  MM      MM    111111      000000    PP          EEEEEEEEEE  NN      NN      ....
RR      RR  MM      MM    111111      000000    PP          EEEEEEEEEE  NN      NN      ....

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

Table of contents

```
0000     1          $BEGIN  RM1OPEN,000,RM$RMS1,<SEQUENTIAL-SPECIFIC OPEN>
0000     2
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     28 ;++
0000     29 ; Facility: RMS32
0000     30 ;
0000     31 ; Abstract:
0000     32 ;
0000     33 ;  this routine performs the sequential file organization-
0000     34 ;  specific open processing.
0000     35 ;
0000     36 ; Environment:
0000     37 ;                star processor running starlet exec.
0000     38 ;
0000     39 ; Author: L f Laverdure,         creation date: 4-JAN-1977
0000     40 ;
0000     41 ; Modified By:
0000     42 ;
0000     43 ;       V03-002 KBT0407         Keith B. Thompson       30-Nov-1982
0000     44 ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz and
0000     45 ;               ifb$w_asdevbsiz to ifb$l_asdevbsiz
0000     46 ;
0000     47 ;       V03-001 KBT0142         Keith B. Thompson       20-Aug-1982
0000     48 ;               Reorganize psects
0000     49 ;
0000     50 ;       V02-015 REFORMAT        Keith B. Thompson       29-Jul-1980
0000     51 ;
0000     52 ;--
0000     53
```

```
                     0000     55          .SBTTL  DECLARATIONS
                     0000     56
                     0000     57 ;
                     0000     58 ; Include Files:
                     0000     59 ;
                     0000     60
                     0000     61 ;
                     0000     62 ; Macros:
                     0000     63 ;
                     0000     64
                     0000     65          $FABDEF
                     0000     66          $DEVDEF
                     0000     67          $IFBDEF
                     0000     68          $RMSDEF
                     0000     69
                     0000     70 ;
                     0000     71 ; Equated Symbols:
                     0000     72 ;
                     0000     73
00000020             0000     74          FOP=FAB$L_FOP*8                 ; bit offset to fop
                     0000     75
                     0000     76 ;
                     0000     77 ; Own Storage:
                     0000     78 ;
                     0000     79
```

```
0000     81          .SBTTL  RMSOPEN1 - SEQUENTIAL OPEN ROUTINE
0000     82
0000     83 ;++
0000     84 ;
0000     85 ;  RMSOPEN1     -
0000     86 ;
0000     87 ;  this routine performs all of the file open functions
0000     88 ;  that are specific to the sequential file organization,
0000     89 ;  including:
0000     90 ;
0000     91 ;       1. checking that write-sharing has not been specified
0000     92 ;       2. forcing var length and rat=FAB$B_RAT for unit record devices
0000     93 ;       3. setting the bls field of the fab to the device buffer size
0000     94 ;       4. performing magtape file positioning
0000     95 ;       5. setting the two close option bits as required
0000     96 ;       6. returning to the common open routine
0000     97 ;
0000     98 ; Calling sequence:
0000     99 ;
0000    100 ;  entered via case branch from RMS$OPEN
0000    101 ;  returns by jumping to RMS$COPRTN
0000    102 ;
0000    103 ; Input Parameters:
0000    104 ;
0000    105 ;       R11     impure area address
0000    106 ;       R9      ifab address
0000    107 ;       R8      fab address
0000    108 ;
0000    109 ; Implicit Inputs:
0000    110 ;
0000    111 ;       the contents of the fab
0000    112 ;
0000    113 ; Output Parameters:
0000    114 ;
0000    115 ;       R0      status code
0000    116 ;       R1 thru R5 destroyed
0000    117 ;
0000    118 ; Implicit Outputs:
0000    119 ;
0000    120 ;       various fields in the ifab and fab are initialized
0000    121 ;
0000    122 ; Completion Codes:
0000    123 ;
0000    124 ;       standard rms in particular, success and shr.
0000    125 ;
0000    126 ; Side Effects:
0000    127 ;
0000    128 ;       none
0000    129 ;
0000    130 ;--
0000    131
```

```
                     0000   133 RMSOPEN1::
                     0000   134         $ISTPT  OPEN1
                     0006   135
                     0006   136 ;
                     0006   137 ;  check that record locking not required
                     0006   138 ;
                     0006   139
05 69    33    E0    0006   140         BBS     #IFB$V_NORECLK,(R9),2$  ; branch if no locking required
         06    E1    000A   141         BBC     #FAB$V_UPI,-            ; error unless user providing
   67 17 A8          000C   142                 FAB$B_SHR(R8),ERRSHR
                     000F   143
                     000F   144 ;
                     000F   145 ; own interlocking
                     000F   146 ;
                     000F   147
                     000F   148 2$:
                     000F   149
                     000F   150 ;
                     000F   151 ;  if a unit record device, force variable length record format
                     000F   152 ;  and record attributes from fab
                     000F   153 ;
                     000F   154
                     000F   155         ASSUME  FAB$C_SEQ EQ 0
                     000F   156         ASSUME  DEV$V_REC EQ 0
   2C 69    E9       000F   157         BLBC    IFB$L_PRIM_DEV(R9),10$  ; branch if not unit rec.
      02    90       0012   158         MOVB    #FAB$C_VAR,-            ; force to variable
   50 A9              0014   159                 IFB$B_RFMORG(R9)
   1E A8    90       0016   160         MOVB    FAB$B_RAT(R8),-         ; and set rat from fab
   51 A9              0019   161                 IFB$B_RAT(R9)
   F8 8F    8B       001B   162         BICB3   #255 ^ <FAB$M_CR!FAB$M_FTN!FAB$M_PRN>,-
51 1E A8              001E   163                 FAB$B_RAT(R8),R1
                     0021   164
                     0021   165 ;
                     0021   166 ; clear all bit carriage control
                     0021   167 ;
                     0021   168
         1B    13    0021   169         BEQL    10$                     ; branch if none speced
      0A 51    02    E1    0023   170         BBC     #FAB$V_PRN,R1,3$        ; branch if prn not speced
      03 1F A8    91    0027   171         CMPB    FAB$B_RFM(R8),#FAB$C_VFC; rfm = vfc?
         50    12    002B   172         BNEQ    ERRRAT                  ; branch if not
         03    90    002D   173         MOVB    #FAB$C_VFC,-            ; change rfm to vfc
      50 A9              002F   174                 IFB$B_RFMORG(R9)
                     0031   175         ASSUME  FAB$V_FTN EQ FAB$V_CR-1
                     0031   176         ASSUME  FAB$V_PRN EQ FAB$V_CR+1
50 51 03 00    EA    0031   177 3$:     FFS     #FAB$V_FTN,#3,R1,R0     ; find the first bit set
                     0036   178         CSB     R0,R1                   ; turn it off
         51    95    003A   179         TSTB    R1                      ; another bit set
         3F    12    003C   180         BNEQ    ERRRAT                  ; branch if so
                     003E   181
                     003E   182 ;
                     003E   183 ;  set bls equal to device buffer size
                     003E   184 ;
                     003E   185
   0094 C9    B0    003E   186 10$:     MOVW    IFB$L_ASDEVBSIZ(R9),-
      3C A8              0042   187                 FAB$W_BLS(R8)
         06    E0    0044   188         BBS     #DEV$V_SPL,-            ; branch if device spooled
      05 69              0046   189                 IFB$L_PRIM_DEV(R9),15$
```

```
                48 A9   B0  0048   190          MOVW    IFB$L_DEVBUFSIZ(R9),-   ; else return real block size
                3C A8       004B   191                  FAB$W_BLS(R8)
                            004D   192 15$:
                            004D   193
                            004D   194 ;
                            004D   195 ;  if device is magtape, then assume eof at max vbn, byte 0
                            004D   196 ;  unless file is write accessed, in which case
                            004D   197 ;  look at nef and if clear then set "at eof" flag
                            004D   198 ;  and declare eof to be at vbn 1, byte 0
                            004D   199 ;  else utilize same positioning as for read access.
                            004D   200 ;
                            004D   201
                    05      E1  004D   202          BBC     #DEV$V_SQD,-            ; branch if not magtape
                1F 69           004F   203                  IFB$L_PRIM_DEV(R9),50$
            74 A9   01      CE  0051   204          MNEGL   #1,IFB$L_EBK(R9)        ; assume not write accessed
                    18      E0  0055   205          BBS     #DEV$V_FOR,-            ; foreign will never be
                14 69           0057   206                  IFB$L_PRIM_DEV(R9),20$
                            0059   207
                            0059   208 ;
                            0059   209 ; positioned at eof
                            0059   210 ;
                            0059   211
                            0059   212          SSB     #IFB$V_ANSI_D,(R9)      ; set ansi flag
            0C 69   30      E1  005D   213          BBC     #IFB$V_WRTACC,(R9),20$  ; branch if true
            08 68   2A      E0  0061   214          BBS     #FAB$V_NEF+FOP,(R8),20$ ; or if nef set
            74 A9   01      D0  0065   215          MOVL    #1,IFB$L_EBK(R9)        ; set eof to vbn 1
                            0069   216          SSB     #IFB$V_EOF,(R9)         ; and set at eof flg
                5C A9   B4  006D   217 20$:     CLRW    IFB$W_FFB(R9)           ; always use byte 0 for mt
                            0070   218
                            0070   219 ;
                            0070   220 ;  return to common open sequence
                            0070   221 ;
                            0070   222
                            0070   223 50$:     RMSSUC                          ; show success
        FF8A'   31          0073   224 OPNRET: BRW     RM$COPRTN
                            0076   225
                            0076   226 ;
                            0076   227 ;  write sharing specified on sequential file
                            0076   228 ;  -  return with error code
                            0076   229 ;
                            0076   230
                            0076   231 ERRSHR:
                            0076   232          RMSERR  SHR
                    05      11  0078   233          BRB     OPNRET1
                            007D   234
                            007D   235 ERRRAT:
                            007D   236          RMSERR  RAT                     ; invalid rat value
                            0082   237 OPNRET1:
        FF7B'   31          0082   238          BRW     RM$COPRTN               ; and get out
                            0085   239
                            0085   240          .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
DEV$V_FOR                       = 00000018
DEV$V_REC                       = 00000000
DEV$V_SPL                       = 00000006
DEV$V_SQD                       = 00000005
ERRRAT                            0000007D R      01
ERRSHR                            00000076 R      01
FAB$B_RAT                       = 0000001E
FAB$B_RFM                       = 0000001F
FAB$B_SHR                       = 00000017
FAB$C_SEQ                       = 00000000
FAB$C_VAR                       = 00000002
FAB$C_VFC                       = 00000003
FAB$L_FOP                       = 00000004
FAB$M_CR                        = 00000002
FAB$M_FTN                       = 00000001
FAB$M_PRN                       = 00000004
FAB$V_CR                        = 00000001
FAB$V_FTN                       = 00000000
FAB$V_NEF                       = 0000000A
FAB$V_PRN                       = 00000002
FAB$V_UPI                       = 00000006
FAB$W_BLS                       = 0000003C
FOP                             = 00000020
IFB$B_RAT                       = 00000051
IFB$B_RFMORG                    = 00000050
IFB$L_ASDEVBSIZ                 = 00000094
IFB$L_DEVBUFSIZ                 = 00000048
IFB$L_EBK                       = 00000074
IFB$L_PRIM_DEV                  = 00000000
IFB$V_ANSI_D                    = 00000026
IFB$V_EOF                       = 00000021
IFB$V_NORECLK                   = 00000033
IFB$V_WRTACC                    = 00000030
IFB$W_FFB                       = 0000005C
OPNRET                            00000073 R      01
OPNRET1                           00000082 R      01
PIO$A_TRACE                       ********   X    01
RM$COPRTN                         ********   X    01
RM$OPEN1                          00000000 RG     01
RMS$_RAT                        = 0001864C
RMS$_SHR                        = 000186B4
TPT$C_OPEN1                       ********   X    01
```

Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS1	00000085 (133.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:01.06
Command processing	117	00:00:00.64	00:00:04.24
Pass 1	242	00:00:05.77	00:00:19.05
Symbol table sort	0	00:00:00.67	00:00:01.22
Pass 2	54	00:00:01.23	00:00:03.63
Symbol table output	6	00:00:00.07	00:00:00.10
Psect synopsis output	2	00:00:00.04	00:00:00.23
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	452	00:00:08.51	00:00:29.53

The working set limit was 1350 pages.
34030 bytes (67 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 658 non-local and 9 local symbols.
240 source lines were read in Pass 1, producing 13 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	15

776 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1OPEN/OBJ=OBJ$:RM1OPEN MSRC$:RM1OPEN/UPDATE=(ENH$:RM1OPEN)+EXECML$/LIB+LIB$:RMS/LIB
```

RM1PUTREC
LIS

RM1PUTSET
LIS

RM1UPDATE
LIS

RM1NXTBLK
LIS

RM1PUTBLD
LIS

RM1RELBLK
LIS

RM1SEQXFR
LIS

RM1PUT
LIS

RM2CONN
LIS

RM1OPEN
LIS

RM1WTLST
LIS

RM1STMFMT
LIS