```
   SSSSSSSSSSS     000000000     RRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
   SSSSSSSSSSS     000000000     RRRRRRRRRRRR    TTTTTTTTTTTTTTT     333333333       222222222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT        333          333  222          222
SSS              000          000  RRR         RRR        TTT                     333                222
SSS              000          000  RRR         RRR        TTT                     333                222
SSS              000          000  RRR         RRR        TTT                     333                222
   SSSSSSSSS     000          000  RRRRRRRRRRR            TTT                333                   222
   SSSSSSSSS     000          000  RRRRRRRRRRRR           TTT                333                   222
   SSSSSSSSS     000          000  RRRRRRRRRRRR           TTT                333                   222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR   RRR              TTT                   333             222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
            SSS  000          000  RRR      RRR           TTT        333        333          222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
SSSSSSSSSSS          000000000     RRR         RRR        TTT           333333333       222222222222222
```

FILEID**SORRMSIO

```
  SSSSSSSS    000000    RRRRRRRR    RRRRRRRR    MM      MM    SSSSSSSS    IIIIII      000000
  SSSSSSSS    000000    RRRRRRRR    RRRRRRRR    MM      MM    SSSSSSSS    IIIIII      000000
SS           00    00   RR     RR   RR     RR   MMMM  MMMM  SS              II       00    00
SS           00    00   RR     RR   RR     RR   MMMM  MMMM  SS              II       00    00
SS           00    00   RR     RR   RR     RR   MM  MM  MM  SS              II       00    00
SS           00    00   RR     RR   RR     RR   MM  MM  MM  SS              II       00    00
  SSSSSS     00    00   RRRRRRRR    RRRRRRRR    MM      MM    SSSSSS        II       00    00
  SSSSSS     00    00   RRRRRRRR    RRRRRRRR    MM      MM    SSSSSS        II       00    00
        SS   00    00   RR  RR      RR  RR      MM      MM          SS      II       00    00
        SS   00    00   RR  RR      RR  RR      MM      MM          SS      II       00    00
        SS   00    00   RR    RR    RR    RR    MM      MM          SS      II       00    00      ....
        SS   00    00   RR    RR    RR    RR    MM      MM          SS      II       00    00      ....
SSSSSSSS      000000    RR     RR   RR     RR   MM      MM    SSSSSSSS    IIIIII      000000      ....
SSSSSSSS      000000    RR     RR   RR     RR   MM      MM    SSSSSSSS    IIIIII      000000      ....

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
```

```
;     1          0001  0 MODULE SOR$RMS_IO (
;     2          0002  0                 IDENT = 'V04-000'       ! File: SORRMSIO.B32 Edit: PDG3026
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**********************************************************************
;     7          0007  1 !*                                                                    *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                            *
;    11          0011  1 !*                                                                    *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                       *
;    18          0018  1 !*                                                                    *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                       *
;    22          0022  1 !*                                                                    *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    25          0025  1 !*                                                                    *
;    26          0026  1 !*                                                                    *
;    27          0027  1 !**********************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 !
;    32          0032  1 ! FACILITY:     VAX-11 SORT/MERGE
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module contains PMS I/O support.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  VAX/VMS user mode
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: Peter D Gilbert, CREATION DATE: 07-Jan-1982
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 !       T03-015         Original
;    45          0045  1 !       T03-016 Set the OFP FOP flag.  Also, if the output file cannot be in
;    46          0046  1 !               print file format, clear the PRN flag.  PDG 13-Dec-1982
;    47          0047  1 !       T03-017 Set COM_MINVFC before calling callback routine in SOR$$OPEN.
;    48          0048  1 !               PDG 20-Dec-1982
;    49          0049  1 !       T03-018 Added protection XAB.  PDG 30-Dec-1982
;    50          0050  1 !       T03-019 Don't allocate UBF unless there are files.  3-Feb-1983
;    51          0051  1 !       T03-020 Don't allow FAB$C_IDX on the $CREATE.  PDG 3-Mar-1983
;    52          0052  1 !       T03-021 Slight change to file protection.  PDG 11-May-1983
;    53          0053  1 !       T03-022 Recover on RMS$_FLK errors on input.  PDG 19-May-1983
;    54          0054  1 !       T03-023 Allow RMS to default protection, then add extra restrictions.
;    55          0055  1 !               PDG 5-Aug-1983
;    56          0056  1 !       T03-024 Law of excluded middle mishap.  Non-fixed-format files are
;    57          0057  1 !               varying.  PDG 15-Aug-1983
```

```
;    58          0058  1 !       T03-025 SOR$$BEST_FILE_NAME assumes NAM$B_RSL and NAM$B_ESL are zero
;    59          0059  1 !               before the OPEN or CREATE.  PDG 10-Nov-1983
;    60          0060  1 !       T03-026 Also set the UPI bit on second $OPEN attempt.  PDG 9-Apr-1984
;    61          0061  1 !--
```

```
;    63          0062  1 LIBRARY 'SYS$LIBRARY:STARLET';
;    64          0063  1 REQUIRE 'SRC$:COM';
;    65          0133  1
;    66          0134  1 FORWARD ROUTINE
;    67          0135  1     CALC_LRL:           CAL_CTXREG,             ! Calc longest record length
;    68          0136  1     SOR$$OPEN:          CAL_CTXREG,             ! Open input and output files
;    69          0137  1     SOR$$RFA_ACCESS:    NOVALUE CAL_ACCESS;     ! Access a record by RFA
;    70          0138  1
;    71          0139  1 EXTERNAL ROUTINE
;    72          0140  1     SOR$$BEST_FILE_NAME:CAL_CTXREG NOVALUE,
;    73          0141  1     SOR$$ALLOCATE:      CAL_CTXREG,             ! Allocate storage
;    74          0142  1     SOR$$ERROR;                                 ! Issue error diagnostics
```

```
;    76          0143  1 ROUTINE CALC_LRL
;    77          0144  1         (
;    78          0145  1         FAB:    REF BLOCK[,BYTE],
;    79          0146  1         FHC:    REF BLOCK[,BYTE]
;    80          0147  1         ):      CAL_CTXREG =
;    81          0148  1 !++
;    82          0149  1 !
;    83          0150  1 ! FUNCTIONAL DESCRIPTION:
;    84          0151  1 !
;    85          0152  1 !       This routine calculates the longest record length of a file
;    86          0153  1 !       based on the information in the FAB and XABs.
;    87          0154  1 !       Note that for VFC format files, this does not include the VFC area.
;    88          0155  1 !
;    89          0156  1 ! FORMAL PARAMETERS:
;    90          0157  1 !
;    91          0158  1 !       FAB.ra.v        Pointer to FAB
;    92          0159  1 !       FHC.ra.v        Pointer to XABFHC
;    93          0160  1 !
;    94          0161  1 ! IMPLICIT INPUTS:
;    95          0162  1 !
;    96          0163  1 !       NONE
;    97          0164  1 !
;    98          0165  1 ! IMPLICIT OUTPUTS:
;    99          0166  1 !
;   100          0167  1 !       NONE
;   101          0168  1 !
;   102          0169  1 ! ROUTINE VALUE:
;   103          0170  1 !
;   104          0171  1 !       The largest record length for this file.  If it can't
;   105          0172  1 !       be determined from the FAB and XAB, returns zero.
;   106          0173  1 !
;   107          0174  1 ! SIDE EFFECTS:
;   108          0175  1 !
;   109          0176  1 !       NONE
;   110          0177  1 !--
;   111          0178  2     BEGIN
;   112          0179  2     LITERAL
;   113          0180  2         BKS_OVER=       24;             ! Bucket overhead for indexed file.
;   114          0181  2
;   115          0182  2     LOCAL
;   116          0183  2         LRL;                            ! Best guess at longest record length.
;   117          0184  2
;   118          0185  2
;   119          0186  2
;   120          0187  2     ! Determine the length of the longest record in the file (not including the
;   121          0188  2     ! VFC area.
;   122          0189  2     !
;   123          0190  2     ! The LRL value does not include the VFC area, unless the file is relative.
;   124          0191  2     ! The MRS includes the VFC area.
;   125          0192  2     ! The BKS and BLS include the VFC area.
;   126          0193  2     !
;   127          0194  2     IF .FHC[XAB$W_LRL] NEQ 0
;   128          0195  2     THEN
;   129          0196  3         BEGIN
;   130          0197  3         LRL = .FHC[XAB$W_LRL];
;   131          0198  3         IF .FAB[FAB$B_ORG] EQL FAB$C_REL
;   132          0199  3         THEN
```

```
0200 3             LRL = .LRL - .FAB[FAB$B_FSZ];
0201 3         END
0202 2     ELIF
0203 2         .FAB[FAB$W_MRS] NEQ 0
0204 2     THEN
0205 2         LRL = .FAB[FAB$W_MRS] - .FAB[FAB$B_FSZ]
0206 2     ELIF
0207 2         .FAB[FAB$B_BKS] NEQ 0
0208 2     THEN
0209 2         LRL = (.FAB[FAB$B_BKS] * COM_K_BPERBLOCK) - BKS_OVER
0210 2     ELSE
0211 2         LRL = .FAB[FAB$W_BLS];
0212 2
0213 2
0214 2     RETURN .LRL;                              ! Return calculated value.
0215 1     END;


                                                        .TITLE  SOR$RMS_IO
                                                        .IDENT  \V04-000\

                                                        .EXTRN  SOR$$BEST_FILE_NAME
                                                        .EXTRN  SOR$$ALLOCATE, SOR$$ERROR

                                                        .PSECT  SOR$RO_CODE,NOWRT,  SHR,  PIC,2

                                    0004 00000 CALC_LRL:
                                                        .WORD   Save R2                      ; 0143
                  51        04  AC  D0 00002            MOVL    FAB, R1                      ; 0198
                  50        08  AC  D0 00006            MOVL    FHC, R0                      ; 0194
                            0A  A0  B5 0000A            TSTW    10(R0)                       ;
                                0B  13 0000D            BEQL    1$                           ;
                  50        0A  A0  3C 0000F            MOVZWL  10(R0), LRL                  ; 0197
                  10        1D  A1  91 00013            CMPB    29(R1), #16                  ; 0198
                                0A  13 00017            BEQL    2$                           ;
                                    04 00019            RET                                  ; 0200
                            36  A1  B5 0001A 1$:        TSTW    54(R1)                       ; 0203
                                0C  13 0001D            BEQL    3$                           ;
                  50        36  A1  3C 0001F            MOVZWL  54(R1), LRL                  ; 0205
                  52        3F  A1  9A 00023 2$:        MOVZBL  63(R1), R2                   ;
                  50            52  C2 00027            SUBL2   R2, LRL                      ;
                                    04 0002A            RET                                  ;
                            3E  A1  95 0002B 3$:        TSTB    62(R1)                       ; 0207
                                0D  13 0002E            BEQL    4$                           ;
                  52        3E  A1  9A 00030            MOVZBL  62(R1), R2                   ; 0209
        52        52            09  78 00034            ASHL    #9, R2, R2                   ;
                  50        E8  A2  9E 00038            MOVAB   -24(R2), LRL                 ;
                                    04 0003C            RET                                  ;
                  50        3C  A1  3C 0003D 4$:        MOVZWL  60(R1), LRL                  ; 0211
                                    04 00041            RET                                  ; 0215

; Routine Size:  66 bytes,    Routine Base:  SOR$RO_CODE + 0000
```

```
;  150     0216  1 GLOBAL ROUTINE SOR$$OPEN
;  151     0217  1         (
;  152     0218  1         LRL_OUT_RTN,            ! Routine to calculate COM_LRL_OUT
;  153     0219  1         LRL_OUT_PRM             ! Parameter to LRL_OUT_RTN
;  154     0220  1         ): CAL_CTXREG =
;  155     0221  1
;  156     0222  1 !++
;  157     0223  1 !
;  158     0224  1 ! FUNCTIONAL DESCRIPTION:
;  159     0225  1 !
;  160     0226  1 !       This routine opens the input file(s) and the output file.
;  161     0227  1 !       It also verifies some attributes of the files.
;  162     0228  1 !
;  163     0229  1 !       Note that the input files are not opened in PASS_FILES.  We delay
;  164     0230  1 !       opening them until after the user has been able to specify whether
;  165     0231  1 !       errors are to be signalled or returned.
;  166     0232  1 !
;  167     0233  1 ! FORMAL PARAMETERS:
;  168     0234  1 !
;  169     0235  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  170     0236  1 !
;  171     0237  1 ! IMPLICIT INPUTS:
;  172     0238  1 !
;  173     0239  1 !       The DDBs for the files have been initialized.
;  174     0240  1 !
;  175     0241  1 ! IMPLICIT OUTPUTS:
;  176     0242  1 !
;  177     0243  1 !       NONE
;  178     0244  1 !
;  179     0245  1 ! ROUTINE VALUE:
;  180     0246  1 !
;  181     0247  1 !       Status code.
;  182     0248  1 !
;  183     0249  1 ! SIDE EFFECTS:
;  184     0250  1 !
;  185     0251  1 !       NONE
;  186     0252  1 !--
;  187     0253  2     BEGIN
;  188     0254  2     EXTERNAL REGISTER
;  189     0255  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  190     0256  2     LOCAL
;  191     0257  2         DDB:    REF DDB_BLOCK,          ! Pointer to DDB for output file
;  192     0258  2         LRL,                            ! Longest record length
;  193     0259  2         TOT_ALQ,                        ! Total allocation quantity
;  194     0260  2         FAB:    $FAB_DECL,              ! FAB block
;  195     0261  2         NAM:    $NAM_DECL VOLATILE,     ! NAM block
;  196     0262  2         FNA:    BLOCK[NAM$C_MAXRSS, BYTE],      ! File name string area
;  197     0263  2         FHC:    BLOCK[XAB$C_FHCLEN, BYTE],      ! File header control block
;  198     0264  2         XABPRO: $XABPRO_DECL,                   ! XAB for file protection
;  199     0265  2         PRO:    WORD,                           ! Protection
;  200     0266  2         STATUS;                 ! Status
;  201     0267  2     LOCAL
;  202     0268  2         WAS_IDX;
;  203     0269  2
;  204     0270  2
;  205     0271  2     ! Initialize the longest record length
;  206     0272  2     !
```

```
;  207          0273  2      LRL = 0;                                ! Start the maximum low
;  208          0274  2
;  209          0275  2
;  210          0276  2      ! Initialize the accumulative input file allocation, using the default
;  211          0277  2      ! for no input files.
;  212          0278  2      !
;  213          0279  2      TOT_ALQ = 0;
;  214          0280  2      IF .CTX[COM_NUM_FILES] EQL 0 THEN TOT_ALQ = DEF_FILE_ALLOC;
;  215          0281  2
;  216          0282  2
;  217          0283  2      ! If the output file is in VFC format, it's FSZ value is computed by:
;  218          0284  2      !   If user specified FSZ,  then the user-specified FSZ
;  219          0285  2      !   Otherwise,              the FSZ of the first input file
;  220          0286  2      !       (if the FSZ of the first input file is 0, RMS will default to 2)
;  221          0287  2      !
;  222          0288  2      ! The storage we require in an internal format node for the VFC area is:
;  223          0289  2      !   For Record sorts: Min( Max(input-FSZ), Max(output-FSZ) )
;  224          0290  2      !   Non-Record sorts: 0 (we don't need the VFC area, or we get it later)
;  225          0291  2      ! If there are no input (output) files, the corresponding FSZ equals 0.
;  226          0292  2      ! This value is computed in CTX[COM_MINVFC].
;  227          0293  2      !
;  228          0294  2      ! The size of the storage we must allocate to hold the VFC area is:
;  229          0295  2      !   If Max(input-FSZ) = 0,  then 0 (and no storage allocated)
;  230          0296  2      !   If Max(output-FSZ) = 0, then 0 (and no storage allocated)
;  231          0297  2      !   Otherwise,              Max( Max(input-FSZ), Max(output-FSZ) )
;  232          0298  2      ! This value is computed in CTX[COM_MAXVFC].
;  233          0299  2      !
;  234          0300  2      ! The calculations are done as follows:
;  235          0301  2      !   Compute Max(input-FSZ) into CTX[COM_MAXVFC]
;  236          0302  2      !
;  237          0303  2      CTX[COM_MAXVFC] = 0;                    ! Start the maximum low
;  238          0304  2
;  239          0305  2
;  240          0306  2      ! Initialize the FAB (file access block), the NAM (name block), and
;  241          0307  2      ! the FHC XAB (file header control extended attributes block).
;  242          0308  2      !
;  243        P 0309  2      $FAB_INIT(
;  244        P 0310  2          FAB = FAB[BASE_],                   ! FAB block
;  245        P 0311  2          NAM = NAM[BASE_],                   ! NAM block
;  246        P 0312  2          XAB = FHC[BASE_],                   ! FHC block
;  247        P 0313  2 !        FNA                                 ! File name area        (set below)
;  248        P 0314  2 !        FNS                                 ! File name area size   (set below)
;  249        P 0315  2          FAC = GET,                          ! File access
;  250        P 0316  2          SHR = GET,                          ! Sharing
;  251        P 0317  2          DNA = UPLIT BYTE(STR_DEF_EXT),      ! Default extension is .DAT
;  252        P 0318  2          DNS = %CHARCOUNT(STR_DEF_EXT),      ! Default extension is .DAT
;  253        P 0319  2          RFM = VAR,                          ! Needed if no input files
;  254          0320  2          RAT = CR);                          ! Record attributes
;  255          0321  2      IF .CTX[COM_SORT_TYPE] NEQ TYP_K_TAG
;  256          0322  2      THEN
;  257          0323  2          FAB[FAB$L_FOP] = FAB$M_SQO;         ! Sequential access only if not tag
;  258        P 0324  2      $NAM_INIT(
;  259        P 0325  2          NAM = NAM[BASE_],                   ! NAM block
;  260        P 0326  2          ESS = %ALLOCATION(FNA),             ! Expanded name string size
;  261        P 0327  2          ESA = FNA[BASE_],                   ! Expanded name string area
;  262        P 0328  2          RSS = %ALLOCATION(FNA),             ! Resultant name string size
;  263          0329  2          RSA = FNA[BASE_]);                  ! Resultant name string area
```

```
;  264        P 0330  2      $XABFHC_INIT(
;  265        P 0331  2          XAB = FHC[BASE_],                   ! XABFHC block
;  266          0332  2          NXT = XABPRO[BASE_]);
;  267          0333  2      PRO = 0;                                ! No protection restrictions yet
;  268          0334  2
;  269          0335  2      ! Loop for each input file
;  270          0336  2      !
;  271          0337  2      DDB = .CTX[COM_INP_DDB];                        ! Point to first DDB
;  272          0338  2      DECR I FROM .CTX[COM_NUM_FILES]-1 TO 0 DO
;  273          0339  3          BEGIN
;  274          0340  3          LOCAL
;  275          0341  3              T;
;  276          0342  3
;  277          0343  3          ! Advance to next DDB.
;  278          0344  3          ! The first input file is opened last, so the output file will use
;  279          0345  3          ! the file characteristics of the first input file.
;  280          0346  3          !
;  281          0347  3          DDB = .DDB[DDB_NEXT];
;  282          0348  3          IF DDB[BASE_] EQL 0 THEN DDB = .CTX[COM_INP_DDB];
;  283          0349  3
;  284          0350  3          $XABPRO_INIT(XAB = XABPRO[BASE_]);
;  285          0351  3
;  286          0352  3          !+
;  287          0353  3          !
;  288          0354  3          ! The following information is needed:
;  289          0355  3          !
;  290          0356  3          ! FAB$B_RFM      Record format
;  291          0357  3          ! FAB$B_FSZ      Length of the VFC area
;  292          0358  3          ! FAB$L_ALQ      File allocation
;  293          0359  3          ! FAB            $OPEN, $CLOSE
;  294          0360  3          ! RAB            $GET
;  295          0361  3          ! RAB            Accessing the file by RFA for tag sorts
;  296          0362  3          ! NAM$B_RSL      Resultant file name string length
;  297          0363  3          ! NAM$L_RSA      Resultant file name string address
;  298          0364  3          ! FHC%.3         Used to calculate the LRL
;  299          0365  3          !
;  300          0366  3          ! Thus, much of the storage may be reclaimed.
;  301          0367  3          !
;  302          0368  3          !-
;  303          0369  3
;  304          0370  3          ! Actually open the input file
;  305          0371  3          !
;  306          0372  3          NAM[NAM$B_RSL] = 0;
;  307          0373  3          NAM[NAM$B_ESL] = 0;
;  308          0374  3          FAB[FAB$W_IFI] = 0;
;  309          0375  3          FAB[FAB$B_FNS] = .VECTOR[ DDB[DDB_NAME], 0 ];
;  310          0376  3          FAB[FAB$L_FNA] = .VECTOR[ DDB[DDB_NAME], 1 ];
;  311          0377  3          STATUS = $OPEN(FAB = FAB[BASE_]);
;  312          0378  3
;  313          0379  3
;  314          0380  3          ! Get the best file name string available
;  315          0381  3          !
;  316          0382  3          SOR$$BEST_FILE_NAME(FAB[BASE_], DDB[DDB_NAME]);
;  317          0383  3
;  318          0384  3          IF .FAB[FAB$L_STS] EQL RMS$_FLK
;  319          0385  3          THEN
;  320          0386  4              BEGIN
```

```
;   321          0387  4              FAB[FAB$B_SHR] = FAB$M_PUT OR FAB$M_GET OR FAB$M_DEL OR FAB$M_UPD
;   322          0388  4                               OR FAB$M_UPI;
;   323          0389  4              FAB[FAB$V_NAM] = TRUE;
;   324          0390  4              FAB[FAB$B_FNS] = .VECTOR[ DDB[DDB_NAME], 0 ];
;   325          0391  4              FAB[FAB$L_FNA] = .VECTOR[ DDB[DDB_NAME], 1 ];
;   326          0392  5              IF $OPEN(FAB = FAB[BASE_])
;   327          0393  4              THEN
;   328          0394  5                  BEGIN
;   329          0395  5                  SOR$$ERROR(
;   330          0396  5                      SOR$_SHR_OPENIN AND NOT STS$M_SEVERITY OR STS$K_WARNING,
;   331          0397  5                      1, DDB[DDB_NAME], RMS$_FLK, 0);
;   332          0398  4                  END;
;   333          0399  4              FAB[FAB$B_SHR] = FAB$M_GET;
;   334          0400  4              FAB[FAB$V_NAM] = FALSE;
;   335          0401  3              END;
;   336          0402  3
;   337          0403  3          IF NOT .FAB[FAB$L_STS]
;   338          0404  3          THEN
;   339          0405  3              RETURN SOR$$ERROR(SOR$_SHR_OPENIN, 1, DDB[DDB_NAME],
;   340          0406  3                      .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
;   341          0407  3
;   342          0408  3
;   343          0409  3          ! If this is not a VFC format file, clear the FSZ field
;   344          0410  3          !
;   345          0411  3          IF .FAB[FAB$B_RFM] NEQ FAB$C_VFC
;   346          0412  3          THEN
;   347          0413  3              FAB[FAB$B_FSZ] = 0;
;   348          0414  3
;   349          0415  3
;   350          0416  3          ! Calculate largest record length
;   351          0417  3          !
;   352          0418  3          T = CALC_LRL(FAB[BASE_], FHC[BASE_]);
;   353          0419  3          IF .LRL EQL 0
;   354          0420  3          THEN
;   355          0421  3              LRL = .T                        ! First time here, just use length
;   356          0422  3          ELIF
;   357          0423  3              .T NEQ .LRL
;   358          0424  3          THEN
;   359          0425  4              BEGIN
;   360          0426  4              IF .T GTRU .LRL THEN LRL = .T;
;   361          0427  4              CTX[COM_VAR] = TRUE;            ! Variable length records
;   362          0428  3              END;
;   363          0429  3
;   364          0430  3
;   365          0431  3          ! Check for VFC format input files.
;   366          0432  3          !
;   367          0433  3          IF .CTX[COM_MAXVFC] LSSU .FAB[FAB$B_FSZ]
;   368          0434  3          THEN
;   369          0435  3              CTX[COM_MAXVFC] = .FAB[FAB$B_FSZ];  ! Maximize COM_MAXVFC
;   370          0436  3
;   371          0437  3
;   372          0438  3          ! Most files are varying in length
;   373          0439  3          !
;   374          0440  3          IF  .FAB[FAB$B_RFM] NEQ FAB$C_FIX
;   375          0441  3          THEN
;   376          0442  3              CTX[COM_VAR] = TRUE;                ! Variable-length records
;   377          0443  3
```

```
;  378          0444  3
;  379          0445  3          ! Get the allocation quantity
;  380          0446  3          ! Note that we naively ignore the complexities of indexed files.
;  381          0447  3          !
;  382          0448  3          IF .BLOCK[ FAB[FAB$L_DEV], DEV$V_RND; ,BYTE]
;  383          0449  3          THEN
;  384          0450  4              BEGIN
;  385          0451  4              !
;  386          0452  4              ! FHC[XAB$L_EBK] should be a better estimate than FAB[FAB$L_ALQ]
;  387          0453  4              !
;  388          0454  4              TOT_ALQ = .TOT_ALQ + .FHC[XAB$L_EBK];
;  389          0455  4              END
;  390          0456  3          ELSE
;  391          0457  4              BEGIN
;  392          0458  4              !
;  393          0459  4              ! The input file is not on a random access device.
;  394          0460  4              !
;  395          0461  4              LOCAL
;  396          0462  4                  ALQ;
;  397          0463  4              IF .CTX[COM_SORT_TYPE] NEQ TYP_K_RECORD
;  398          0464  4              THEN
;  399          0465  4                  RETURN SOR$_BAD_TYPE;   ! Only random access devices have RFAs
;  400          0466  4              IF (ALQ = .FHC[XAB$L_EBK]) EQL 0 THEN
;  401          0467  4              IF (ALQ = .FAB[FAB$L_ALQ]) EQL 0 THEN
;  402          0468  4              IF .BLOCK[ FAB[FAB$L_DEV], DEV$V_TRM; ,BYTE] THEN
;  403          0469  4                  ALQ = DEF_TRM_ALLOC
;  404          0470  4              ELSE
;  405          0471  4                  ALQ = DEF_FILE_ALLOC;
;  406          0472  4              TOT_ALQ = .TOT_ALQ + .ALQ;
;  407          0473  3              END;
;  408          0474  3
;  409          0475  3
;  410        P 0476  3          $RAB_INIT(
;  411        P 0477  3              RAB = DDB[DDB_RAB+BASE_],
;  412        P 0478  3              FAB = FAB[BASE_],
;  413        P 0479  3 !            MBC                     ! May be set below
;  414        P 0480  3 !            MBF                     ! Set below
;  415        P 0481  3              RAC = SEQ,
;  416        P 0482  3 !            RHB =                   ! Allocated later
;  417          0483  3              ROP = <RAH,LOC,MAS>);
;  418          0484  3
;  419          0485  3
;  420          0486  3          ! If organization is sequential and the device is disk use MBC and MBF
;  421          0487  3          ! if there are more than 8 blocks available.  Otherwise use MBF = 2.
;  422          0488  3          !
;  423          0489  3          ! ??? Is this the best way to calculate these values?
;  424          0490  3          !
;  425          0491  3          IF .FAB[FAB$B_ORG] NEQ FAB$C_SEQ OR
;  426          0492  3              .BLOCK[ FAB[FAB$L_DEV], DEV$V_SQD; ,BYTE] OR
;  427          0493  3              NOT .BLOCK[ FAB[FAB$L_DEV], DEV$V_RND; ,BYTE]
;  428          0494  3          THEN
;  429          0495  3              DDB[DDB_RAB+RAB$B_MBF] = MAX_MBF
;  430          0496  3          ELSE
;  431          0497  4              BEGIN
;  432          0498  4              DDB[DDB_RAB+RAB$B_MBC] = MAX_MBC;
;  433          0499  4              DDB[DDB_RAB+RAB$B_MBF] = MAX_MBF;
;  434          0500  3              END;
```

```
;  435          0501  3
;  436          0502  3          ! Connect the RAB to the FAB
;  437          0503  3          !
;  438          0504  3          STATUS = $CONNECT(RAB = DDB[DDB_RAB+BASE_]);
;  439          0505  3          IF NOT .STATUS
;  440          0506  3          THEN
;  441          0507  3              RETURN SOR$$ERROR(SOR$_SHR_OPENIN, 1, DDB[DDB_NAME],
;  442          0508  3                      .DDB[DDB_RAB+RAB$L_STS], .DDB[DDB_RAB+RAB$L_STV]);
;  443          0509  3
;  444          0510  3          ! Make the protection even more prohibitive,
;  445          0511  3          !
;  446          0512  3          PRO = .PRO OR .XABPRO[XAB$W_PRO];
;  447          0513  3
;  448          0514  3          ! Save the IFI and FOP
;  449          0515  3          !
;  450          0516  3          DDB[DDB_IFI] = .FAB[FAB$W_IFI];
;  451          0517  3          DDB[DDB_FOP] = .FAB[FAB$L_FOP];
;  452          0518  2          END;
;  453          0519  2
;  454          0520  2
;  455          0521  2      ! Store the LRL value into the common context area.
;  456          0522  2      ! If the LRL was specified by the user, use that.
;  457          0523  2      ! If the LRL was not specified, use the value from the input files.
;  458          0524  2      ! Check the value of the LRL.
;  459          0525  2      ! Note that we do allow a calculated LRL to be zero.
;  460          0526  2      !
;  461          0527  2      IF .CTX[COM_LRL] NEQ 0                          ! Did the user specify a value?
;  462          0528  2      THEN
;  463          0529  2          0                                           ! Yes, leave it alone
;  464          0530  2      ELSE
;  465          0531  3          BEGIN
;  466          0532  3          CTX[COM_LRL] = .LRL;                        ! No, use our value
;  467          0533  3          IF .LRL GTRU MAX_REFSIZE
;  468          0534  3          THEN
;  469          0535  3              RETURN SOR$$ERROR(SOR$_LRL_MISS);
;  470          0536  2          END;
;  471          0537  2
;  472          0538  2
;  473          0539  2      ! Allocate space for the user buffer, and set the UBF and USZ.
;  474          0540  2      !
;  475          0541  2      IF .CTX[COM_NUM_FILES] NEQ 0
;  476          0542  2      THEN
;  477          0543  3          BEGIN
;  478          0544  3          LOCAL
;  479          0545  3              USZ,
;  480          0546  3              UBF:        REF BLOCK;
;  481          0547  3          USZ = .CTX[COM_LRL];    ! + .CTX[COM_MAXVFC];
;  482          0548  3          UBF = SOR$$ALLOCATE(.USZ);
;  483          0549  3          DDB = .CTX[COM_INP_DDB];
;  484          0550  3          DECR I FROM .CTX[COM_NUM_FILES]-1 TO 0 DO
;  485          0551  4              BEGIN
;  486          0552  4              DDB[DDB_RAB+RAB$W_USZ] = .USZ;
;  487          0553  4              DDB[DDB_RAB+RAB$L_UBF] = UBF[BASE_];
;  488          0554  4              DDB = .DDB[DDB_NEXT];
;  489          0555  3              END;
;  490          0556  2          END;
;  491          0557  2
```

```
;  492          0558  2
;  493          0559  2      ! Figure the number of blocks needed to store all the input records.
;  494          0560  2      !
;  495          0561  2      IF .CTX[COM_FILE_ALLOC] NEQ 0
;  496          0562  2      THEN
;  497          0563  2          0                                   ! User told us; assume he knows best
;  498          0564  2      ELSE
;  499          0565  2          CTX[COM_FILE_ALLOC] = .TOT_ALQ; ! Use the input file allocation
;  500          0566  2
```

```
;  502          0567  2      ! If no output file is specified, update the VFC values appropriately.
;  503          0568  2      !
;  504          0569  2      DDB = .CTX[COM_OUT_DDB];
;  505          0570  2      IF DDB[BASE_] EQL 0
;  506          0571  2      THEN
;  507          0572  3          BEGIN
;  508          0573  3          !
;  509          0574  3          ! Max(output-FSZ) = 0
;  510          0575  3          ! CTX[COM_MINVFC] = Min( Max(input-FSZ), Max(output-FSZ) ) = 0
;  511          0576  3          ! CTX[COM_MAXVFC] = 0   (no storage needed for this)
;  512          0577  3          !
;  513          0578  3          CTX[COM_MINVFC] = CTX[COM_MAXVFC] = 0;
;  514          0579  2          END;
;  515          0580  2
;  516          0581  2      ! The size we need in internal nodes, COM_MINVFC, may be needed by the
;  517          0582  2      ! the routine we are about to call.  Set it pessimisticly (since we don't
;  518          0583  2      ! know about the output file yet).
;  519          0584  2      !
;  520          0585  2      CTX[COM_MINVFC] = .CTX[COM_MAXVFC];
;  521          0586  2
;  522          0587  2
;  523          0588  2      ! Now that we know the longest input record length, set the largest output
;  524          0589  2      ! record length.  Record reformatting, and the sort process determine the
;  525          0590  2      ! output record length, so call a routine to calculate COM_LRL_OUT.
;  526          0591  2      !
;  527          0592  2      STATUS = CAL_CTXREG(.LRL_OUT_RTN, .LRL_OUT_PRM);
;  528          0593  2      IF NOT .STATUS THEN RETURN .STATUS;
;  529          0594  2
;  530          0595  2
;  531          0596  2      !+
;  532          0597  2      !
;  533          0598  2      ! The only fields in the context area that are set or modified below are:
;  534          0599  2      ! COM_LRL_OUT, COM_MINVFC, and COM_MAXVFC
;  535          0600  2      !
;  536          0601  2      ! COM_LRL_OUT may be modified to hold the maximum record size for fixed
;  537          0602  2      ! format output files, so that, if a record length occurs when writing a
;  538          0603  2      ! record, we have a correct length that can be used.
;  539          0604  2      !
;  540          0605  2      !-
;  541          0606  2
;  542          0607  2
;  543          0608  2      ! If no output file is specified, return now.
;  544          0609  2      !
;  545          0610  2      IF DDB[BASE_] EQL 0 THEN RETURN SS$_NORMAL;
;  546          0611  2
;  547          0612  2  !+
;  548          0613  2  !
;  549          0614  2  ! Fall through here only if an output file was specified                                ; I
;  550          0615  2  !
;  551          0616  2  ! The following values (computed above) are used:
;  552          0617  2  !       LRL     Longest record length
;  553          0618  2  !       TOT_ALQ Total input file allocation
;  554          0619  2  !       VFC     Size of fixed portion of VFC records
;  555          0620  2  !-
;  556          0621  2
;  557          0622  2      ! Initialize the FAB for output
;  558          0623  2      !
```

```
;  559          0624  2      FAB[FAB$W_IFI] = 0;
;  560          0625  2      FAB[FAB$B_FAC] = FAB$M_PUT;
;  561          0626  2      FAB[FAB$B_SHR] = FAB$M_NIL;
;  562          0627  2      FAB[FAB$B_FNS] = .VECTOR[ DDB[DDB_NAME], 0 ];
;  563          0628  2      FAB[FAB$L_FNA] = .VECTOR[ DDB[DDB_NAME], 1 ];
;  564          0629  2      FHC[XAB$W_LRL] = 0;
;  565          0630  2
;  566          0631  2      ! Set the output file protection, requesting that RMS tell us what it used.
;  567          0632  2      !
;  568          0633  2      $XABPRO_INIT(XAB = XABPRO[BASE_]);
;  569          0634  2      XABPRO[XAB$W_PRO] = -1;
;  570          0635  2
;  571          0636  2      ! Initialize the Record Access Block
;  572          0637  2      !
;  573        P 0638  2      $RAB_INIT(
;  574        P 0639  2          RAB = DDB[DDB_RAB+BASE_],
;  575        P 0640  2          FAB = FAB[BASE_],
;  576        P 0641  2 !        MBC                 ! May be set below
;  577        P 0642  2 !        MBF                 ! Set below
;  578        P 0643  2          RAC = SEQ,
;  579        P 0644  2 !        RHB                 ! Allocated later
;  580          0645  2          ROP = <WBH,MAS>);
;  581          0646  2      IF .CTX[COM_LOAD_FILL] THEN DDB[DDB_RAB+RAB$V_LOA] = TRUE;
;  582          0647  2
;  583          0648  2
;  584          0649  2      ! The ALQ field is used to preallocate a file when it is created.
;  585          0650  2      ! This saves on the number of extends needed when creating the file,
;  586          0651  2      ! and helps ensure that sufficient space will be available for the
;  587          0652  2      ! output file.  However, this may decrease the amount of space available
;  588          0653  2      ! for work files, and may be inaccurate due to record selection, or INDEX
;  589          0654  2      ! or ADDRESS sorts.
;  590          0655  2      !
;  591        L 0656  2      %IF TUN_K_OUT_PREALL
;  592          0657  2      %THEN
;  593          0658  2          FAB[FAB$L_ALQ] = .TOT_ALQ;
;  594          0659  2      %FI
;  595          0660  2
;  596          0661  2
;  597          0662  2      ! Default the maximum record size now, and allow the user to override it.
;  598          0663  2      !
;  599          0664  2      ! Delay opening the output file until the keys, et.al have been processed,
;  600          0665  2      ! because of record reformatting.
;  601          0666  2      !
;  602          0667  2      FAB[FAB$W_MRS] = %X'FFFF';          ! Indicate MRS is uninitialized
;  603          0668  2
;  604          0669  2
;  605          0670  2      ! If address or index sort, default organization to sequential and record
;  606          0671  2      ! format to fixed.  Allow RMS to default block and bucket size.
;  607          0672  2      ! The longest output record length was calculated by the LRL_OUT_RTN.
;  608          0673  2      !
;  609          0674  3      IF ONEOF_(.CTX[COM_SORT_TYPE], BMSK_(TYP_K_ADDRESS,TYP_K_INDEX))
;  610          0675  2      THEN
;  611          0676  3          BEGIN
;  612          0677  3          FAB[FAB$B_ORG] = FAB$C_SEQ;             ! Sequential organization
;  613          0678  3          FAB[FAB$B_RFM] = FAB$C_FIX;             ! Fixed length records
;  614          0679  3          FAB[FAB$B_RAT] = FAB$M_CR;              ! So we can look at it
;  615          0680  2          END;
```

```
;  616          0681  2
;  617          0682  2
;  618          0683  2      ! Set file options.
;  619          0684  2      ! By default, we want to trucate at the end of file, unless the user
;  620          0685  2      ! has explicitly specified an output file allocation, or if the user
;  621          0686  2      ! has specified file options to be used.
;  622          0687  2      ! TEF = truncate at end of file
;  623          0688  2      !
;  624          0689  2      FAB[FAB$L_FOP] = .FAB[FAB$L_FOP] OR FAB$M_TEF;
;  625          0690  2
;  626          0691  2
;  627          0692  2      ! Copy user-specified output file options into the FAB.
;  628          0693  2      !
;  629          0694  2      IF .CTX[COM_PASS_FILES] NEQ 0
;  630          0695  2      THEN
;  631          0696  3          BEGIN
;  632          0697  3          LOCAL
;  633          0698  3              P:  REF VECTOR;
;  634          0699  3          P = .CTX[COM_PASS_FILES];
;  635          0700  3          IF .(.P)<1,1> THEN FAB[FAB$B_ORG] = .P[1];
;  636          0701  3          IF .(.P)<2,1> THEN FAB[FAB$B_RFM] = .P[2];
;  637          0702  3          IF .(.P)<3,1> THEN FAB[FAB$B_BKS] = .P[3];
;  638          0703  3          IF .(.P)<4,1> THEN FAB[FAB$W_BLS] = .P[4];
;  639          0704  3          IF .(.P)<5,1> THEN FAB[FAB$W_MRS] = .P[5];
;  640          0705  4          IF .(.P)<6,1> THEN BEGIN
;  641          0706  4                             FAB[FAB$L_ALQ] = .P[6];
;  642          0707  4                             FAB[FAB$V_TEF] = FALSE;
;  643          0708  3                             END;
;  644          0709  3          IF .(.P)<7,1> THEN FAB[FAB$L_FOP] = .P[7];
;  645          0710  3          IF .(.P)<8,1> THEN FAB[FAB$B_FSZ] = .P[8];
;  646          0711  2          END;
;  647          0712  2
;  648          0713  2
;  649          0714  2      ! Set other file options.
;  650          0715  2      ! We want to use deferred writes, regardless of what the user specified.
;  651          0716  2      ! DFW = deferred write
;  652          0717  2      ! SQO = sequential access only
;  653          0718  2      ! OFP = output file parse
;  654          0719  2      !
;  655          0720  2      FAB[FAB$L_FOP] = .FAB[FAB$L_FOP] OR FAB$M_DFW OR FAB$M_SQO OR FAB$M_OFP;
;  656          0721  2
;  657          0722  2
;  658          0723  2      ! If the user did not specify an MRS value, default it as needed.
;  659          0724  2      !
;  660          0725  2      IF .FAB[FAB$W_MRS] EQL %X'FFFF'
;  661          0726  2      THEN
;  662          0727  3          BEGIN
;  663          0728  3
;  664          0729  3          ! If relative or fixed format, we must set MRS.
;  665          0730  3          !
;  666          0731  3          ! Remember that MRS includes the length of the VFC area
;  667          0732  3          !
;  668          0733  3          IF .FAB[FAB$B_ORG] EQL FAB$C_REL OR .FAB[FAB$B_RFM] EQL FAB$C_FIX
;  669          0734  3          THEN
;  670          0735  4              BEGIN
;  671          0736  4              LOCAL
;  672          0737  4                  FSZ;
```

```
;  673          0738  4              FAB[FAB$W_MRS] = .CTX[COM_LRL_OUT];
;  674          0739  4              FSZ = .FAB[FAB$B_FSZ];
;  675          0740  4              IF .FSZ EQL 0 THEN FSZ = 2;          ! RMS default
;  676          0741  4              IF .FAB[FAB$B_RFM] EQL FAB$C_VFC
;  677          0742  4              THEN
;  678          0743  4                  FAB[FAB$W_MRS] = .FAB[FAB$W_MRS] + .FSZ;
;  679          0744  4              END
;  680          0745  3          ELSE
;  681          0746  3              FAB[FAB$W_MRS] = 0;
;  682          0747  3
;  683          0748  2          END;
;  684          0749  2
;  685          0750  2      WAS_IDX = FALSE;
;  686          0751  2      IF .FAB[FAB$B_ORG] EQL FAB$C_IDX
;  687          0752  2      THEN
;  688          0753  3          BEGIN
;  689          0754  3          IF NOT .FAB[FAB$V_CIF]
;  690          0755  3          THEN
;  691          0756  4              BEGIN
;  692          0757  4              !
;  693          0758  4              ! We seem to be creating an indexed output file.
;  694          0759  4              ! Complain and change the organization.
;  695          0760  4              !
;  696          0761  4              SOR$$ERROR(SOR$_IND_OVR AND NOT STS$M_SEVERITY OR STS$K_WARNING);
;  697          0762  4              END
;  698          0763  3          ELSE
;  699          0764  4              BEGIN
;  700          0765  4              !
;  701          0766  4              ! Remember that the caller expects to overlay an indexed file.
;  702          0767  4              ! Default the organization.  If the file is created (and is not
;  703          0768  4              ! indexed), complain.
;  704          0769  4              !
;  705          0770  4              WAS_IDX = TRUE;
;  706          0771  3              END;
;  707          0772  3
;  708          0773  3          ! Default the organization
;  709          0774  3          !
;  710          0775  3          FAB[FAB$B_ORG] = 0;
;  711          0776  2          END;
;  712          0777  2
;  713          0778  2
;  714          0779  2      ! Print file format files must be VFC with FSZ of at least 2
;  715          0780  2      !
;  716          0781  2      IF .FAB[FAB$B_RFM] NEQ FAB$C_VFC OR .FAB[FAB$B_FSZ] LSS 2
;  717          0782  2      THEN
;  718          0783  2          FAB[FAB$V_PRN] = FALSE;
;  719          0784  2
;  720          0785  2
;  721          0786  2      ! Create the output file
;  722          0787  2      !
;  723          0788  3      BEGIN
;  724          0789  3      LOCAL
;  725          0790  3          ONAM:   $NAM_DECL;
;  726          0791  3
;  727        P 0792  3      $NAM_INIT(
;  728        P 0793  3          NAM = ONAM[BASE_],              ! NAM block
;  729        P 0794  3          ESS = %ALLOCATION(FNA),         ! Expanded name string size
```

```
;  730        P 0795  3          ESA = FNA[BASE_],                       ! Expanded name string area
;  731        P 0796  3          RSS = %ALLOCATION(FNA),                 ! Resultant name string size
;  732          0797  3          RSA = FNA[BASE_]);                      ! Resultant name string area
;  733          0798  3
;  734          0799  3      FAB[FAB$L_NAM] = ONAM[BASE_];
;  735          0800  3
;  736          0801  3
;  737          0802  3      ! Use the first input file as a related file name string
;  738          0803  3      !
;  739          0804  3      IF .CTX[COM_NUM_FILES] NEQ 0
;  740          0805  3      THEN
;  741          0806  4          BEGIN
;  742          0807  4          ONAM[NAM$L_RLF] = NAM[BASE_];
;  743          0808  4          FAB[FAB$B_DNS] = 0;                     ! Get rid of the default name string
;  744          0809  4          FAB[FAB$L_DNA] = 0;                     ! Get rid of the default name string
;  745          0810  3          END;
;  746          0811  3
;  747          0812  3      ! Create the output file.
;  748          0813  3      !
;  749          0814  3      ! Note that we are unwilling to do many checks on the file attributes,
;  750          0815  3      ! since RMS is good at doing that.  Also, any checks that are done should
;  751          0816  3      ! be done after the create, since the specified file attributes may not be
;  752          0817  3      ! the same as the actual attributes (due to the CIF option, and defaults).
;  753          0818  3      !
;  754          0819  3      STATUS = $CREATE(FAB = FAB[BASE_]);
;  755          0820  3
;  756          0821  3      ! Get the best file name string available.
;  757          0822  3      !
;  758          0823  3      SOR$$BEST_FILE_NAME(FAB[BASE_], DDB[DDB_NAME]);
;  759          0824  3
;  760          0825  2      END,
;  761          0826  2
;  762          0827  2
;  763          0828  2      IF .WAS_IDX AND .FAB[FAB$L_STS] EQL RMS$_CREATED
;  764          0829  2      THEN
;  765          0830  3          BEGIN
;  766          0831  3          !
;  767          0832  3          ! Oops.  We created a sequential file instead of an indexed file.
;  768          0833  3          ! Inform the caller.
;  769          0834  3          !
;  770          0835  3          SOR$$ERROR(SOR$_IND_OVR AND NOT STS$M_SEVERITY OR STS$K_WARNING);
;  771          0836  2          END;
;  772          0837  2
;  773          0838  2
;  774          0839  2      IF NOT .FAB[FAB$L_STS]
;  775          0840  2      THEN
;  776          0841  2          RETURN SOR$$ERROR(SOR$_SHR_OPENOUT, 1, DDB[DDB_NAME],
;  777          0842  2                  .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
;  778          0843  2
;  779          0844  2
;  780          0845  2      ! If we really created the file, check the protection
;  781          0846  2      !
;  782          0847  2      IF NOT .FAB[FAB$V_CIF] OR .FAB[FAB$L_STS] EQL RMS$_CREATED
;  783          0848  2      THEN
;  784          0849  3          BEGIN
;  785          0850  3
;  786          0851  3          ! Verify that the protection is as restrictive as we want it to be.
```

```
;  787          0852  3          ! Leave owner, delete and write protections alone, since we're only
;  788          0853  3          ! interested in prohibiting processes that couldn't read the original
;  789          0854  3          ! files.  If the protection is not resrictive enough, change it.
;  790          0855  3          !
;  791          0856  3          LOCAL
;  792          0857  3              CHANGE_MASK: WORD;          ! Bits we will want to change
;  793          0858  3          LITERAL
;  794          0859  3              M_RELEVANT = %X'5505';      ! W:DEWR,G:DEWR,O:DEWR,S:DEWR
;  795          0860  3          EXTERNAL ROUTINE
;  796          0861  3              LIB$SET_FILE_PROT:  ADDRESSING_MODE(GENERAL);
;  797          0862  3          EXTERNAL LITERAL
;  798          0863  3              LIB$_INVFILSPE;             ! Invalid file spec, or file not on disk
;  799          0864  3
;  800          0865  3          CHANGE_MASK = NOT .XABPRO[XAB$W_PRO] AND .PRO AND M_RELEVANT;
;  801          0866  3          IF .CHANGE_MASK NEQ 0
;  802          0867  3          THEN
;  803          0868  4              BEGIN
;  804          0869  4              STATUS = LIB$SET_FILE_PROT(
;  805          0870  4                  DDB[DDB_NAME],                  ! File specification string
;  806          0871  4                  CHANGE_MASK,                    ! Mask of bits to change
;  807          0872  4                  PRO);                           ! Mask of bit values
;  808          0873  4              IF NOT .STATUS AND .STATUS NEQ LIB$_INVFILSPE
;  809          0874  4              THEN
;  810          0875  4                  RETURN SOR$$ERROR(
;  811          0876  4                      SOR$_SHR_OPENOUT AND NOT STS$M_SEVERITY OR STS$K_WARNING,
;  812          0877  4                      1, DDB[DDB_NAME], .STATUS);
;  813          0878  3              END;
;  814          0879  2          END;
;  815          0880  2
;  816          0881  2      ! If this is not a VFC format file, clear the FSZ field
;  817          0882  2      ! (since RMS does not clear it).
;  818          0883  2      !
;  819          0884  2      IF .FAB[FAB$B_RFM] NEQ FAB$C_VFC
;  820          0885  2      THEN
;  821          0886  2          FAB[FAB$B_FSZ] = 0;
;  822          0887  2
;  823          0888  2
;  824          0889  2      ! Adjust the longest output record length
;  825          0890  2      !
;  826          0891  2      IF .FAB[FAB$W_MRS] EQL 0
;  827          0892  2      THEN
;  828          0893  2          0       ! The only restriction is due to physical limitations.
;  829          0894  2      ELSE
;  830          0895  3          BEGIN
;  831          0896  3          !
;  832          0897  3          ! Set the output LRL to the record length for the file.
;  833          0898  3          ! Thus, we have the correct output length available.
;  834          0899  3          !
;  835          0900  3 !        IF .FAB[FAB$B_RFM] EQL FAB$C_FIX
;  836          0901  3 !        THEN
;  837          0902  3              CTX[COM_LRL_OUT] = .FAB[FAB$W_MRS] - .FAB[FAB$B_FSZ];
;  838          0903  2          END;
;  839          0904  2
;  840          0905  2
;  841          0906  2      ! More VFC processing
;  842          0907  2      !
;  843          0908  2      ! Remember, COM_MINVFC is the size we need in internal nodes,
```

```
;  844     0909  2      ! and COM_MAXVFC is the size we need to allocate for RMS.
;  845     0910  2      !
;  846     0911  2      CTX[COM_MINVFC] = MINU( .CTX[COM_MAXVFC], .FAB[FAB$B_FSZ] );
;  847     0912  2      IF .CTX[COM_MINVFC] EQL 0
;  848     0913  2      THEN
;  849     0914  2          CTX[COM_MAXVFC] = 0     ! No storage needed for this
;  850     0915  2      ELSE
;  851     0916  2          CTX[COM_MAXVFC] = MAXU( .CTX[COM_MAXVFC], .FAB[FAB$B_FSZ] );
;  852     0917  2      IF .CTX[COM_SORT_TYPE] NEQ TYP_K_RECORD
;  853     0918  2      THEN
;  854     0919  2          CTX[COM_MINVFC] = 0;    ! Not needed in the nodes
;  855     0920  2
;  856     0921  2
;  857     0922  2      !+
;  858     0923  2      !
;  859     0924  2      ! Various checks are not made.
;  860     0925  2      !
;  861     0926  2      ! Do not check converting variable-length input to fixed-length output.
;  862     0927  2      !
;  863     0928  2      ! If the file was overlaid, do not check that user-specified attributes
;  864     0929  2      ! agree with the files existing attributes.
;  865     0930  2      !
;  866     0931  2      ! Don't check for creating an indexed file (with an awful primary key),
;  867     0932  2      ! since RMS won't create an indexed file unless a KEY XAB is used.
;  868     0933  2      !
;  869     0934  2      ! Don't check that the output of an address or index sort is really
;  870     0935  2      ! sequential and fixed-format.
;  871     0936  2      !
;  872     0937  2      !-
;  873     0938  2
;  874     0939  2
;  875     0940  2      ! If the file was not created, and the file is not empty,
;  876     0941  2      ! set the EOF option to position to the end-of-file before writing records.
;  877     0942  2      ! Note that the EOF option is only allowed for sequential files.  Thus,
;  878     0943  2      ! for sequential files, the records will be appended to the file,
;  879     0944  2      ! for relative files, the records will be appended to the file,
;  880     0945  2      ! for indexed files, mass-insert gives better performance.
;  881     0946  2      !
;  882     0947  2      ! If this is removed, an error occurs for sequential and relative files.
;  883     0948  2      ! We may do this so that the user will not get unexpected results, and to
;  884     0949  2      ! avoid any effects of the NEF and POS file options.
;  885     0950  2      !
;  886     0951  2      ! P.S. If we can't insert records in an indexed file sequentially, we will
;  887     0952  2      ! switch over to keyed inserts.
;  888     0953  2      !
;  889     0954  2      IF .FAB[FAB$V_CIF] AND .FAB[FAB$L_STS] NEQ RMS$_CREATED
;  890     0955  2      THEN
;  891     0956  2          IF .FAB[FAB$B_ORG] NEQ FAB$C_IDX
;  892     0957  2          THEN
;  893     0958  2              DDB[DDB_RAB+RAB$V_EOF] = TRUE;
;  894     0959  2
;  895     0960  2
;  896     0961  2      ! If organization is sequential and the device is disk use MBC and MBF
;  897     0962  2      ! if there are more than 8 blocks available.  Otherwise use MBF = 2.
;  898     0963  2      !
;  899     0964  2      IF .FAB[FAB$B_ORG] NEQ FAB$C_SEQ OR
;  900     0965  2          .BLOCK[ FAB[FAB$L_DEV], DEV$V_SQD; ,BYTE] OR
```

```
;  901          0966  2          NOT .BLOCK[ FAB[FAB$L_DEV], DEV$V_RND; ,BYTE]
;  902          0967  2      THEN
;  903          0968  2          DDB[DDB_RAB+RAB$B_MBF] = MAX_MBF
;  904          0969  2      ELSE
;  905          0970  3          BEGIN
;  906          0971  3          DDB[DDB_RAB+RAB$B_MBC] = MAX_MBC;
;  907          0972  3          DDB[DDB_RAB+RAB$B_MBF] = MAX_MBF;
;  908          0973  2          END;
;  909          0974  2
;  910          0975  2
;  911          0976  2      ! Connect to the FAB
;  912          0977  2      !
;  913          0978  2      STATUS = $CONNECT(RAB = DDB[DDB_RAB+BASE_]);
;  914          0979  2      IF NOT .STATUS
;  915          0980  2      THEN
;  916          0981  2          RETURN SOR$$ERROR(SOR$_SHR_OPENOUT, 1, DDB[DDB_NAME],
;  917          0982  2                  .DDB[DDB_RAB+RAB$L_STS], .DDB[DDB_RAB+RAB$L_STV]);
;  918          0983  2
;  919          0984  2
;  920          0985  2      ! Save the IFI and FOP
;  921          0986  2      !
;  922          0987  2      DDB[DDB_IFI] = .FAB[FAB$W_IFI];
;  923          0988  2      DDB[DDB_FOP] = .FAB[FAB$L_FOP];
;  924          0989  2
;  925          0990  2      RETURN SS$_NORMAL;
;  926          0991  1      END;


                                    54  41  44  2E  00042 P.AAA:  .ASCII  \.DAT\                                          ;

                                                                  .EXTRN  SYS$OPEN, SYS$CONNECT
                                                                  .EXTRN  SYS$CREATE, LIB$SET_FILE_PROT
                                                                  .EXTRN  LIB$_INVFILSPE

                                        07FC 00000                .ENTRY  SOR$$OPEN, Save R2,R3,R4,R5,R6,R7,R8,R9,R10 ; 0216
                                5E    FD60  CE  9E 00002          MOVAB   -672(SP), SP                                    ;
                                      59    D4 00007              CLRL    LRL                                       ; 0273
                                      7E    D4 00009              CLRL    TOT_ALQ                                   ; 0279
                                        59  AB 95 0000B          TSTB    89(CTX)                                   ; 0280
                                            05 12 0000E          BNEQ    1$                                        ;
                                6E    0180  8F 3C 00010          MOVZWL  #384, TOT_ALQ                             ;
                                      0082  CB 94 00015 1$:      CLRB    130(CTX)                                  ; 0303
0050  8F             00          6E        00 2C 00019          MOVC5   #0, (SP), #0, #80, $RMS_PTR               ; 0320
                                    B0      AD    00020
                          B0  AD  5003      8F B0 00022          MOVW    #20483, $RMS_PTR                          ;
                          C6  AD  0202      8F B0 00028          MOVW    #514, $RMS_PTR+22                         ;
                          CE  AD  0202      8F B0 0002E          MOVW    #514, $RMS_PTR+30                         ;
                          D4  AD  00C8      CE 9E 00034          MOVAB   FHC, $RMS_PTR+36                          ;
                          D8  AD  FF50      CD 9E 0003A          MOVAB   NAM, $RMS_PTR+40                          ;
                          E0  AD    B9      AF 9E 00040          MOVAB   P.AAA, $RMS_PTR+48                        ;
                          E5  AD            04 90 00045          MOVB    #4, $RMS_PTR+53                           ;
                                02    58    AB 91 00049          CMPB    88(CTX), #2                               ; 0321
                                            05 13 0004D          BEQL    2$                                        ;
                          B4  AD      40    8F 9A 0004F          MOVZBL  #64, FAB+4                                ; 0323
0060  8F             00          6E        00 2C 00054 2$:      MOVC5   #0, (SP), #0, #96, $RMS_PTR               ; 0329
                                    FF50  CD    0005B
```

```
                        FF50  CD   6002  8F  B0 0005E       MOVW    #24578, $RMS_PTR                   :
                        FF52  CD         01  8E 00065       MNEGB   #1, $RMS_PTR+2                     :
                        FF54  CD   00F4  CE  9E 0006A       MOVAB   FNA, $RMS_PTR+4                    :
                        FF5A  CD         01  8E 00071       MNEGB   #1, $RMS_PTR+10                    :
                        FF5C  CD   00F4  CE  9E 00076       MOVAB   FNA, $RMS_PTR+12                   :
      2C         00           6E         00  2C 0007D       MOVC5   #0, (SP), #0, #44, $RMS_PTR        : 0332
                                   00C8  CE     00082                                                  :
                        00C8  CE   2C1D  8F  B0 00085       MOVW    #11293, $RMS_PTR                   :
                        00CC  CE     70  AE  9E 0008C       MOVAB   XABPRO, $RMS_PTR+4                 :
                                     08  AE  B4 00092       CLRW    PRO                                : 0333
                              57   009C  CB  D0 00095       MOVL    156(CTX), DDB                      : 0337
                              5A     59  AB  9A 0009A       MOVZBL  89(CTX), I                         : 0338
                                       0189  31 0009E       BRW     22$                                :
                              57         67  D0 000A1 3$:   MOVL    (DDB), DDB                         : 0347
                                         05  12 000A4       BNEQ    4$                                 : 0348
                              57   009C  CB  D0 000A6       MOVL    156(CTX), DDB                      :
 0058  8F        00           6E         00  2C 000AB 4$:   MOVC5   #0, (SP), #0, #88, $RMS_PTR        : 0350
                                     70  AE     000B2                                                  :
                          70  AE   5813  8F  B0 000B4       MOVW    #22547, $RMS_PTR                   :
                                   FF53  CD  94 000BA       CLRB    NAM+3                              : 0372
                                   FF5B  CD  94 000BE       CLRB    NAM+11                             : 0373
                                     B2  AD  B4 000C2       CLRW    FAB+2                              : 0374
                              58     04  A7  9E 000C5       MOVAB   4(DDB), R8                         : 0375
                          E4  AD         68  90 000C9       MOVB    (R8), FAB+52                       :
                          DC  AD     04  A8  D0 000CD       MOVL    4(R8), FAB+44                      : 0376
                                     B0  AD  9F 000D2       PUSHAB  FAB                                : 0377
                   00000000G  00         01  FB 000D5       CALLS   #1, SYS$OPEN                       :
                          04  AE         50  D0 000DC       MOVL    R0, STATUS                         :
                                         58  DD 000E0       PUSHL   R8                                 : 0382
                                     B0  AD  9F 000E2       PUSHAB  FAB                                :
                   00000000G  00         02  FB 000E5       CALLS   #2, SOR$$BEST_FILE_NAME            :
                   0001828A   8F     B8  AD  D1 000EC       CMPL    FAB+8, #98954                      : 0384
                                         40  12 000F4       BNEQ    6$                                 :
                          C7  AD     4F  8F  90 000F6       MOVB    #79, FAB+23                        : 0388
                          B7  AD         01  88 000FB       BISB2   #1, FAB+7                          : 0389
                          E4  AD         68  90 000FF       MOVB    (R8), FAB+52                       : 0390
                          DC  AD     04  A8  D0 00103       MOVL    4(R8), FAB+44                      : 0391
                                     B0  AD  9F 00108       PUSHAB  FAB                                : 0392
                   00000000G  00         01  FB 0010B       CALLS   #1, SYS$OPEN                       :
                              19         50  E9 00112       BLBC    R0, 5$                             :
                                         7E  D4 00115       CLRL    -(SP)                              : 0397
                               0001828A  8F  DD 00117       PUSHL   #98954                             :
                                         58  DD 0011D       PUSHL   R8                                 :
                                         01  DD 0011F       PUSHL   #1                                 :
                               001C1098  8F  DD 00121       PUSHL   #1839256                           :
                   00000000G  00         05  FB 00127       CALLS   #5, SOR$$ERROR                     :
                          C7  AD         02  90 0012E 5$:   MOVB    #2, FAB+23                         : 0399
                          B7  AD         01  8A 00132       BICB2   #1, FAB+7                          : 0400
                              07     B8  AD  E8 00136 6$    BLBS    FAB+8, 7$                          : 0403
                              7E     B8  AD  7D 0013A       MOVQ    FAB+8, -(SP)                       : 0406
                                       00CD  31 0013E       BRW     20$                                : 0405
                              03     CF  AD  91 00141 7$:   CMPB    FAB+31, #3                         : 0411
                                         03  13 00145       BEQL    8$                                 :
                                     EF  AD  94 00147       CLRB    FAB+63                             : 0413
                                   00C8  CE  9F 0014A 8$:   PUSHAB  FHC                                : 0418
                                     B0  AD  9F 0014E       PUSHAB  FAB                                :
                        FE64  CF         02  FB 00151       CALLS   #2, CALC_LRL                       :
```

```
                                      59  D5 00156         TSTL    LRL                              ; 0419
                                      05  12 00158         BNEQ    9$
                        59            50  D0 0015A         MOVL    T, LRL                           ; 0421
                                      0F  11 0015D         BRB     11$
                        59            50  D1 0015F 9$:     CMPL    T, LRL                           ; 0423
                                      0A  13 00162         BEQL    11$
                                      03  1B 00164         BLEQU   10$                              ; 0426
                        59            50  D0 00166         MOVL    T, LRL
                  0080  CB            02  88 00169 10$:    BISB2   #2, 128(CTX)                     ; 0427
                        50      0080  CB  9E 0016E 11$:    MOVAB   128(CTX), R0                     ; 0433
                    EF  AD        02  A0  91 00173         CMPB    2(R0), FAB+63
                                      05  1E 00178         BGEQU   12$
                    02  A0        EF  AD  90 0017A         MOVB    FAB+63, 2(R0)                    ; 0435
                        01        CF  AD  91 0017F 12$:    CMPB    FAB+31, #1                       ; 0440
                                      03  13 00183         BEQL    13$
                        60            02  88 00185         BISB2   #2, (R0)                         ; 0442
            07      F3  AD            04  E1 00188 13$:    BBC     #4, FAB+67, 14$                  ; 0448
                        6E      00D8  CE  C0 0018D         ADDL2   FHC+16, TOT_ALQ                  ; 0454
                                      2D  11 00192         BRB     18$                              ; 0448
                        01        58  AB  91 00194 14$:    CMPB    88(CTX), #1                      ; 0463
                                      08  13 00198         BEQL    15$
                        50 001C806C   8F  D0 0019A         MOVL    #1867884, R0                     ; 0465
                                          04 001A1         RET
                        50      00D8  CE  D0 001A2 15$:    MOVL    FHC+16, ALQ                      ; 0466
                                      15  12 001A7         BNEQ    17$
                        50        C0  AD  D0 001A9         MOVL    FAB+16, ALQ                      ; 0467
                                      0F  12 001AD         BNEQ    17$
            05      F0  AD            02  E1 001AF         BBC     #2, FAB+64, 16$                  ; 0468
                        50            10  D0 001B4         MOVL    #16, ALQ                         ; 0469
                                      05  11 001B7         BRB     17$
                        50      0180  8F  3C 001B9 16$:    MOVZWL  #384, ALQ                        ; 0471
                        6E            50  C0 001BE 17$:    ADDL2   ALQ, TOT_ALQ                     ; 0472
                        56        14  A7  9E 001C1 18$:    MOVAB   20(DDB), R6                      ; 0483
0044  8F    00          6E            00  2C 001C5         MOVC5   #0, (SP), #0, #68, (R6)
                                      66     001CC
                        66      4401  8F  B0 001CD         MOVW    #17409, (R6)
                    04  A6  00010220  8F  D0 001D2         MOVL    #66080, 4(R6)
                                  1E  A6  94 001DA         CLRB    30(R6)
                    3C  A6        B0  AD  9E 001DD         MOVAB   FAB, 60(R6)
                                  CD  AD  95 001E2         TSTB    FAB+29                           ; 0491
                                      0E  12 001E5         BNEQ    19$
            09      F0  AD            05  E0 001E7         BBS     #5, FAB+64, 19$                  ; 0492
            04      F3  AD            04  E1 001EC         BBC     #4, FAB+67, 19$                  ; 0493
                    4B  A7            10  90 001F1         MOVB    #16, 75(DDB)                     ; 0498
                    4A  A7            02  90 001F5 19$:    MOVB    #2, 74(DDB)                      ; 0499
                                      56  DD 001F9         PUSHL   R6                               ; 0504
             00000000G  00            01  FB 001FB         CALLS   #1, SYS$CONNECT
                    04  AE            50  D0 00202         MOVL    R0, STATUS
                        11        04  AE  E8 00206         BLBS    STATUS, 21$                      ; 0505
                        7E        1C  A7  7D 0020A         MOVQ    28(DDB), -(SP)                   ; 0508
                                      58  DD 0020E 20$:    PUSHL   R8                               ; 0507
                                      01  DD 00210         PUSHL   #1
                          001C109C    8F  DD 00212         PUSHL   #1839260
                                    0348  31 00218         BRW     67$
                    08  AE        78  AE  A8 0021B 21$:    BISW2   XABPRO+8, PRO                    ; 0512
                    0C  A7        B2  AD  3C 00220         MOVZWL  FAB+2, 12(DDB)                   ; 0516
                    10  A7        B4  AD  D0 00225         MOVL    FAB+4, 16(DDB)                   ; 0517
```

```
                         02          5A  F4 0022A 22$:   SOBGEQ  I, 23$                                   ; 0338
                                     03  11 0022D        BRB     24$                                      ;
                                   FE6F  31 0022F 23$:   BRW     3$                                       ;
                         52    0084  CB  9E 00232 24$:   MOVAB   132(CTX), R2                             ; 0527
                                     62  B5 00237        TSTW    (R2)                                     ;
                                     1A  12 00239        BNEQ    25$                                      ;
                         62          59  B0 0023B        MOVW    LRL, (R2)                                ; 0532
            0000FFFF     8F          59  D1 0023E        CMPL    LRL, #65535                              ; 0533
                                     0E  1B 00245        BLEQU   25$                                      ;
                             001C8074  8F  DD 00247        PUSHL   #1867892                                 ; 0535
            00000000G    00          01  FB 0024D        CALLS   #1, SOR$$ERROR                           ;
                                         04 00254        RET                                              ;
                               59    AB  95 00255 25$:   TSTB    89(CTX)                                  ; 0541
                                     25  13 00258        BEQL    28$                                      ;
                         52          62  3C 0025A        MOVZWL  (R2), USZ                                ; 0547
                                     52  DD 0025D        PUSHL   USZ                                      ; 0548
            00000000G    00          01  FB 0025F        CALLS   #1, SOR$$ALLOCATE                        ;
                         57    009C  CB  D0 00266        MOVL    156(CTX), DDB                            ; 0549
                         51      59  AB  9A 0026B        MOVZBL  89(CTX), I                               ; 0553
                                     0B  11 0026F        BRB     27$                                      ;
                    34   A7          52  B0 00271 26$:   MOVW    USZ, 52(DDB)                             ; 0552
                    38   A7          50  D0 00275        MOVL    UBF, 56(DDB)                             ; 0553
                         57          67  D0 00279        MOVL    (DDB), DDB                               ; 0554
                         F2          51  F4 0027C 27$:   SOBGEQ  I, 26$                                   ; 0550
                               00A8  CB  D5 0027F 28$:   TSTL    168(CTX)                                 ; 0561
                                     05  12 00283        BNEQ    29$                                      ;
                  00A8   CB          6E  D0 00285        MOVL    TOT_ALQ, 168(CTX)                        ; 0565
                         57    0098  CB  D0 0028A 29$:   MOVL    152(CTX), DDB                            ; 0569
                                     52  D4 0028F        CLRL    R2                                       ; 0570
                                     57  D5 00291        TSTL    DDB                                      ;
                                     06  12 00293        BNEQ    30$                                      ;
                                     52  D6 00295        INCL    R2                                       ;
                               0081  CB  B4 00297        CLRW    129(CTX)                                 ; 0578
                  0081   CB    0082  CB  90 0029B 30$:   MOVB    130(CTX), 129(CTX)                       ; 0585
                                 08  AC  DD 002A2        PUSHL   LRL_OUT_PRM                              ; 0592
                    04   BC          01  FB 002A5        CALLS   #1, @LRL_OUT_RTN                         ;
                    04   AE          50  D0 002A9        MOVL    R0, STATUS                               ;
                         05      04  AE  E8 002AD        BLBS    STATUS, 31$                              ; 0593
                         50      04  AE  D0 002B1        MOVL    STATUS, R0                               ;
                                         04 002B5        RET                                              ;
                         03          52  E9 002B6 31$:   BLBC    R2, 32$                                  ; 0610
                                   02B9  31 002B9        BRW     69$                                      ;
                                 B2  AD  B4 002BC 32$:   CLRW    FAB+2                                    ; 0624
                    C6   AD    2001  8F  B0 002BF        MOVW    #8193, FAB+22                            ; 0625
                         5A      04  A7  9E 002C5        MOVAB   4(DDB), R10                              ; 0627
                    E4   AD          6A  90 002C9        MOVB    (R10), FAB+52                            ;
                    DC   AD      04  AA  D0 002CD        MOVL    4(R10), FAB+44                           ; 0628
                               00D2  CE  B4 002D2        CLRW    FHC+10                                   ; 0629
  0058     8F         00             6E  00  2C 002D6        MOVC5   #0, (SP), #0, #88, $RMS_PTR              ; 0633
                                 70  AE     002DD                                                         ;
                    70   AE    5813  8F  B0 002DF        MOVW    #22547, $RMS_PTR                         ;
                    78   AE          01  AE 002E5        MNEGW   #1, XABPRO+8                             ; 0634
                         56      14  A7  9E 002E9        MOVAB   20(DDB), R6                              ; 0645
  0044     8F         00             6E  00  2C 002ED        MOVC5   #0, (SP), #0, #68, (R6)                  ;
                                     66     002F4                                                         ;
                         66    4401  8F  B0 002F5        MOVW    #17409, (R6)                             ;
                    04   A6    0420  8F  3C 002FA        MOVZWL  #1056, 4(R6)                             ;
```

```
                          1E  A6  94 00300        CLRB    30(R6)                              ;
         3C     A6        B0  AD  9E 00303        MOVAB   FAB, 60(R6)                         ;
04       5B     AB            04  E1 00308        BBC     #4, 91(CTX), 33$                    ; 0646
         19     A7            20  88 0030D        BISB2   #32, 25(DDB)                        ;
         C0     AD            6E  D0 00311 33$:   MOVL    TOT_ALQ, FAB+16                     ; 0658
         E6     AD            01  AE 00315        MNEGW   #1, FAB+54                          ; 0667
50 18000000     8F        58  AB  78 00319        ASHL    88(CTX), #402653184, R0             ; 0674
                              0A  18 00322        BGEQ    34$                                 ;
         CF     AD            01  90 00324        MOVB    #1, FAB+31                          ; 0678
         CD     AD      0200  8F  B0 00328        MOVW    #512, FAB+29                        ; 0677
         B7     AD            10  88 0032E 34$:   BISB2   #16, FAB+4                          ; 0689
                50      0094  CB  D0 00332        MOVL    148(CTX), R0                        ; 0694
                              4C  13 00337        BEQL    42$                                 ;
05              60            01  E1 00339        BBC     #1, (P), 35$                        ; 0700
         CD     AD        04  A0  90 0033D        MOVB    4(P), FAB+29                        ;
05              60            02  E1 00342 35$:   BBC     #2, (P), 36$                        ; 0701
         CF     AD        08  A0  90 00346        MOVB    8(P), FAB+31                        ;
05              60            03  E1 0034B 36$:   BBC     #3, (P), 37$                        ; 0702
         EE     AD        0C  A0  90 0034F        MOVB    12(P), FAB+62                       ;
05              60            04  E1 00354 37$:   BBC     #4, (P), 38$                        ; 0703
         EC     AD        10  A0  B0 00358        MOVW    16(P), FAB+60                       ;
05              60            05  E1 0035D 38$:   BBC     #5, (P), 39$                        ; 0704
         E6     AD        14  A0  B0 00361        MOVW    20(P), FAB+54                       ;
09              60            06  E1 00366 39$:   BBC     #6, (P), 40$                        ; 0705
         C0     AD        18  A0  D0 0036A        MOVL    24(P), FAB+16                       ; 0706
         B7     AD            10  8A 0036F        BICB2   #16, FAB+7                          ; 0707
                              60  95 00373 40$:   TSTB    (P)                                 ; 0709
                              05  18 00375        BGEQ    41$                                 ;
         B4     AD        1C  A0  D0 00377        MOVL    28(P), FAB+4                        ;
                05        01  A0  E9 0037C 41$:   BLBC    1(P), 42$                           ; 0710
         EF     AD        20  A0  90 00380        MOVB    32(P), FAB+63                       ;
         B4     AD  20000060  8F  C8 00385 42$:   BISL2   #536871008, FAB+4                   ; 0720
       FFFF     8F        E6  AD  B1 0038D        CMPW    FAB+54, #65535                      ; 0725
                              2A  12 00393        BNEQ    46$                                 ;
                10        CD  AD  91 00395        CMPB    FAB+29, #16                         ; 0733
                              06  13 00399        BEQL    43$                                 ;
                01        CF  AD  91 0039B        CMPB    FAB+31, #1                          ;
                              1B  12 0039F        BNEQ    45$                                 ;
         E6     AD      008A  CB  B0 003A1 43$:   MOVW    138(CTX), FAB+54                    ; 0738
                50        EF  AD  9A 003A7        MOVZBL  FAB+63, FSZ                         ; 0739
                              03  12 003AB        BNEQ    44$                                 ; 0740
                50            02  D0 003AD        MOVL    #2, FSZ                             ;
                03        CF  AD  91 003B0 44$:   CMPB    FAB+31, #3                          ; 0741
                              09  12 003B4        BNEQ    46$                                 ;
         E6     AD            50  A0 003B6        ADDW2   FSZ, FAB+54                         ; 0743
                              03  11 003BA        BRB     46$                                 ; 0733
                          E6  AD  B4 003BC 45$:   CLRW    FAB+54                              ; 0746
                              59  D4 003BF 46$:   CLRL    WAS_IDX                             ; 0750
                20        CD  AD  91 003C1        CMPB    FAB+29, #32                         ; 0751
                              1A  12 003C5        BNEQ    49$                                 ;
0F       B7     AD            01  E0 003C7        BBS     #1, FAB+7, 47$                      ; 0754
                    001C8050  8F  DD 003CC        PUSHL   #1867856                            ; 0761
   00000000G    00            01  FB 003D2        CALLS   #1, SOR$$ERROR                      ;
                              03  11 003D9        BRB     48$                                 ; 0754
                59            01  D0 003DB 47$:   MOVL    #1, WAS_IDX                         ; 0770
                          CD  AD  94 003DE 48$:   CLRB    FAB+29                              ; 0775
                03        CF  AD  91 003E1 49$:   CMPB    FAB+31, #3                          ; 0781
```

```
                                          06  12 003E5          BNEQ    50$                                    ;
                           02         EF  AD  91 003E7          CMPB    FAB+63, #2                             ;
                                          04  1E 003EB          BGEQU   51$                                    ;
                      CE   AD             04  8A 003ED 50$:     BICB2   #4, FAB+30                             ; 0783
  0060   8F           00   6E             00  2C 003F1 51$:     MOVC5   #0, (SP), #0, #96, $RMS_PTR            ; 0797
                                      10  AE     003F8                                                         ;
                      10   AE       6002  8F  B0 003FA          MOVW    #24578, $RMS_PTR                       ;
                      12   AE             01  8E 00400          MNEGB   #1, $RMS_PTR+2                         ;
                      14   AE       00F4  CE  9E 00404          MOVAB   FNA, $RMS_PTR+4                        ;
                      1A   AE             01  8E 0040A          MNEGB   #1, $RMS_PTR+10                        ;
                      1C   AE       00F4  CE  9E 0040E          MOVAB   FNA, $RMS_PTR+12                       ;
                      D8   AD         10  AE  9E 00414          MOVAB   ONAM, FAB+40                           ; 0799
                                      59  AB  95 00419          TSTB    89(CTX)                                ; 0804
                                          0C  13 0041C          BEQL    52$                                    ;
                      20   AE       FF50  CD  9E 0041E          MOVAB   NAM, ONAM+16                           ; 0807
                                      E5  AD  94 00424          CLRB    FAB+53                                 ; 0808
                                      E0  AD  D4 00427          CLRL    FAB+48                                 ; 0809
                                      B0  AD  9F 0042A 52$:     PUSHAB  FAB                                    ; 0819
               00000000G   00             01  FB 0042D          CALLS   #1, SYS$CREATE                         ;
                      04   AE             50  D0 00434          MOVL    R0, STATUS                             ;
                                          5A  DD 00438          PUSHL   R10                                    ; 0823
                                      B0  AD  9F 0043A          PUSHAB  FAB                                    ;
               00000000G   00             02  FB 0043D          CALLS   #2, SOR$$BEST_FILE_NAME                ;
                           17             59  E9 00444          BLBC    WAS_IDX, 53$                           ; 0828
               00010619    8F         B8  AD  D1 00447          CMPL    FAB+8, #67097                          ;
                                          0D  12 0044F          BNEQ    53$                                    ;
                               001C8050   8F  DD 00451          PUSHL   #1867856                               ; 0835
               00000000G   00             01  FB 00457          CALLS   #1, SOR$$ERROR                         ;
                           07         B8  AD  E8 0045E 53$:     BLBS    FAB+8, 54$                             ; 0839
                           7E         B8  AD  7D 00462          MOVQ    FAB+8, -(SP)                           ; 0842
                                        00F0  31 00466          BRW     66$                                    ; 0841
            0A        B7   AD             01  E1 00469 54$:     BBC     #1, FAB+7, 55$                         ; 0847
               00010619    8F         B8  AD  D1 0046E          CMPL    FAB+8, #67097                          ;
                                          4A  12 00476          BNEQ    56$                                    ;
                           50         08  AE  3C 00478 55$:     MOVZWL  PRO, R0                                ; 0865
                           51         78  AE  3C 0047C          MOVZWL  XABPRO+8, R1                           ;
                           50             51  CA 00480          BICL2   R1, R0                                 ;
        0C   AE            50       AAFA  8F  AB 00483          BICW3   #-21766, R0, CHANGE_MASK               ;
                                          36  13 0048A          BEQL    56$                                    ; 0866
                                      08  AE  9F 0048C          PUSHAB  PRO                                    ; 0870
                                      10  AE  9F 0048F          PUSHAB  CHANGE_MASK                            ;
                                          5A  DD 00492          PUSHL   R10                                    ;
               00000000G   00             03  FB 00494          CALLS   #3, LIB$SET_FILE_PROT                  ;
                      04   AE             50  D0 0049B          MOVL    R0, STATUS                             ;
                           1F         04  AE  E8 0049F          BLBS    STATUS, 56$                            ; 0873
               00000000G   8F         04  AE  D1 004A3          CMPL    STATUS, #LIB$_INVFILSPE                ;
                                          15  13 004AB          BEQL    56$                                    ;
                                      04  AE  DD 004AD          PUSHL   STATUS                                 ; 0877
                                          5A  DD 004B0          PUSHL   R10                                    ;
                                          01  DD 004B2          PUSHL   #1                                     ;
                               001C10A0   8F  DD 004B4          PUSHL   #1839264                               ;
               00000000G   00             04  FB 004BA          CALLS   #4, SOR$$ERROR                         ;
                                              04 004C1          RET                                            ;
                           03         CF  AD  91 004C2 56$:     CMPB    FAB+31, #3                             ; 0884
                                          03  13 004C6          BEQL    57$                                    ;
                                      EF  AD  94 004C8          CLRB    FAB+63                                 ; 0886
                                      E6  AD  B5 004CB 57$:     TSTW    FAB+54                                 ; 0891
```

```
                                      0B  13 004CE         BEQL    58$                            ;
                        50       EF   AD  9A 004D0         MOVZBL  FAB+63, R0                     ; 0902
     008A   CB      E6  AD            50  A3 004D4         SUBW3   R0, FAB+54, 138(CTX)           ;
                        50     0080   CB  9E 004DB 58$:    MOVAB   128(CTX), R0                   ; 0911
                        51       02   A0  9A 004E0         MOVZBL  2(R0), R1                      ;
                        51       EF   AD  91 004E4         CMPB    FAB+63, R1                     ;
                                      04  1E 004E8         BGEQU   59$                            ;
                        51       EF   AD  9A 004EA         MOVZBL  FAB+63, R1                     ;
                    01  A0            51  90 004EE 59$:    MOVB    R1, 1(R0)                      ;
                                      05  12 004F2         BNEQ    60$                            ; 0912
                                 02   A0  94 004F4         CLRB    2(R0)                          ; 0914
                                      12  11 004F7         BRB     62$                            ;
                        51       02   A0  9A 004F9 60$:    MOVZBL  2(R0), R1                      ; 0916
                        51       EF   AD  91 004FD         CMPB    FAB+63, R1                     ;
                                      04  1B 00501         BLEQU   61$                            ;
                        51       EF   AD  9A 00503         MOVZBL  FAB+63, R1                     ;
                    02  A0            51  90 00507 61$:    MOVB    R1, 2(R0)                      ;
                        01       58   AB  91 0050B 62$:    CMPB    88(CTX), #1                    ; 0917
                                      03  13 0050F         BEQL    63$                            ;
                                 01   A0  94 00511         CLRB    1(R0)                          ; 0919
               14       B7  AD        01  E1 00514 63$:    BBC     #1, FAB+7, 64$                 ; 0954
           00010619     8F       B8   AD  D1 00519         CMPL    FAB+8, #67097                  ;
                                      0A  13 00521         BEQL    64$                            ;
                        20       CD   AD  91 00523         CMPB    FAB+29, #32                    ; 0956
                                      04  13 00527         BEQL    64$                            ;
                    19  A7            01  88 00529         BISB2   #1, 25(DDB)                    ; 0958
                                 CD   AD  95 0052D 64$:    TSTB    FAB+29                         ; 0964
                                      0E  12 00530         BNEQ    65$                            ;
               09       F0  AD        05  E0 00532         BBS     #5, FAB+64, 65$                ; 0965
               04       F3  AD        04  E1 00537         BBC     #4, FAB+67, 65$                ; 0966
                        4B  A7        10  90 0053C         MOVB    #16, 75(DDB)                   ; 0971
                        4A  A7        02  90 00540 65$:    MOVB    #2, 74(DDB)                    ; 0972
                                      56  DD 00544         PUSHL   R6                             ; 0978
           00000000G    00            01  FB 00546         CALLS   #1, SYS$CONNECT                ;
                    04  AE            50  D0 0054D         MOVL    R0, STATUS                     ;
                        16       04   AE  E8 00551         BLBS    STATUS, 68$                    ; 0979
                        7E       1C   A7  7D 00555         MOVQ    28(DDB), -(SP)                 ; 0982
                                      5A  DD 00559 66$:    PUSHL   R10                            ; 0981
                                      01  DD 0055B         PUSHL   #1                             ;
                       001C10A4       8F  DD 0055D         PUSHL   #1839268                       ;
           00000000G    00            05  FB 00563 67$:    CALLS   #5, SOR$$ERROR                 ;
                                          04 0056A         RET                                    ;
                    0C  A7       B2   AD  3C 0056B 68$:    MOVZWL  FAB+2, 12(DDB)                 ; 0987
                    10  A7       B4   AD  D0 00570         MOVL    FAB+4, 16(DDB)                 ; 0988
                        50            01  D0 00575 69$:    MOVL    #1, R0                         ; 0990
                                          04 00578         RET                                    ; 0991

; Routine Size:  1401 bytes,    Routine Base:  SOR$RO_CODE + 0046
```

```
;  928     0992  1 GLOBAL ROUTINE SOR$$RFA_ACCESS
;  929     0993  1         (
;  930     0994  1         RFA:    REF BLOCK[RAB$S_RFA,BYTE];      ! Addr of the RFA
;  931     0995  1         LEN,                                    ! Length of record
;  932     0996  1         ADR                                     ! Address of record
;  933     0997  1         ):      NOVALUE CAL_ACCESS =
;  934     0998  1
;  935     0999  1 !++
;  936     1000  1 !
;  937     1001  1 ! FUNCTIONAL DESCRIPTION:
;  938     1002  1 !
;  939     1003  1 !       This routine accesses a record by RFA, which is already in the RAB.
;  940     1004  1 !
;  941     1005  1 ! FORMAL PARAMETERS:
;  942     1006  1 !
;  943     1007  1 !       RFA.raw.r       Address of the RFA, possibly follwed by a file number
;  944     1008  1 !       LEN.waw.r       Address of returned length
;  945     1009  1 !       ADR.wal.r       Address of returned address
;  946     1010  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  947     1011  1 !
;  948     1012  1 ! IMPLICIT INPUTS:
;  949     1013  1 !
;  950     1014  1 !       The DDB for the input file.
;  951     1015  1 !
;  952     1016  1 ! IMPLICIT OUTPUTS:
;  953     1017  1 !
;  954     1018  1 !       NONE
;  955     1019  1 !
;  956     1020  1 ! ROUTINE VALUE:
;  957     1021  1 !
;  958     1022  1 !       Status code.
;  959     1023  1 !
;  960     1024  1 ! SIDE EFFECTS:
;  961     1025  1 !
;  962     1026  1 !       NONE
;  963     1027  1 !--
;  964     1028  2     BEGIN
;  965     1029  2     EXTERNAL REGISTER
;  966     1030  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK;
;  967     1031  2     LOCAL
;  968     1032  2         DDB:    REF DDB_BLOCK,
;  969     1033  2         STATUS;
;  970     1034  2
;  971     1035  2
;  972     1036  2     ! Determine whether the RFA is immediately followed by a file number.
;  973     1037  2     ! If so (because there is more than one input file), grab the DDB from the
;  974     1038  2     ! array of DDBs, otherwise, just use the first (only) input DDB.
;  975     1039  2     !
;  976     1040  2     IF .CTX[COM_NUM_FILES] LEQ 1
;  977     1041  2     THEN
;  978     1042  2         DDB = .CTX[COM_INP_DDB]
;  979     1043  2     ELSE
;  980     1044  2         ASSERT_(COM_ORD_FILE EQL COM_ORD_RFA+1)
;  981     1045  2         DDB = .VECTOR[.CTX[COM_INP_ARRAY], .RFA[RAB$S_RFA,0,8,0]];
;  982     1046  2
;  983     1047  2
;  984     1048  2     ASSERT_(RAB$S_RFA EQL 6)
```

```
;  985          1049  2
;  986          1050  2       DDB[DDB_RAB+RAB$L_RFA0] = .RFA[0,0,32,0];     ! Copy the RFA
;  987          1051  2       DDB[DDB_RAB+RAB$W_RFA4] = .RFA[4,0,16,0];
;  988          1052  2
;  989          1053  2       STATUS = $GET(RAB = DDB[DDB_RAB+BASE_]);     ! Read from the file
;  990          1054  2       IF NOT .STATUS
;  991          1055  2       THEN
;  992          1056  2           SOR$$ERROR(SOR$_SHR_READERR, 1, DDB[DDB_NAME],
;  993          1057  2               .DDB[DDB_RAB+RAB$L_STS], .DDB[DDB_RAB+RAB$L_STV]);
;  994          1058  2
;  995          1059  2       LEN = .DDB[DDB_RAB+RAB$W_RSZ];
;  996          1060  2       ADR = .DDB[DDB_RAB+RAB$L_RBF];
;  997          1061  2
;  998          1062  1       END;


                                                          .EXTRN  SYS$GET

                                      0004 00000          .ENTRY  SOR$$RFA_ACCESS, Save R2                    ; 0992
                01      59    AB   91 00002               CMPB    89(CTX), #1                                 ; 1040
                              07   1A 00006               BGTRU   1$                                          ;
                52    009C    CB   D0 00008               MOVL    156(CTX), DDB                               ; 1042
                              10   11 0000D               BRB     2$                                          ;
                50      04    AC   D0 0000F 1$:           MOVL    RFA, R0                                     ; 1045
                50            06   C0 00013               ADDL2   #6, R0                                      ;
                50            60   9A 00016               MOVZBL  (R0), R0                                    ;
                52    00A4  DB40   D0 00019               MOVL    @164(CTX)[R0], DDB                          ;
                50      04    AC   D0 0001F 2$:           MOVL    RFA, R0                                     ; 1050
           24   A2            60   D0 00023               MOVL    (R0), 36(DDB)                               ;
           28   A2      04    A0   B0 00027               MOVW    4(R0), 40(DDB)                              ; 1051
                        14    A2   9F 0002C               PUSHAB  20(DDB)                                     ; 1053
      00000000G 00            01   FB 0002F               CALLS   #1, SYS$GET                                 ;
                16            50   E8 00036               BLBS    STATUS, 3$                                  ; 1054
                7E      1C    A2   7D 00039               MOVQ    28(DDB), -(SP)                              ; 1057
                        04    A2   9F 0003D               PUSHAB  4(DDB)                                      ; 1056
                              01   DD 00040               PUSHL   #1                                          ;
                  001C10B2    8F   DD 00042               PUSHL   #1839282                                    ;
      00000000G 00            05   FB 00048               CALLS   #5, SOR$$ERROR                              ;
                50      36    A2   3C 0004F 3$:           MOVZWL  54(DDB), LEN                                ; 1059
                51      3C    A2   D0 00053               MOVL    60(DDB), ADR                                ; 1060
                                   04 00057               RET                                                 ; 1062

; Routine Size:  88 bytes,    Routine Base:  SOR$RO_CODE + 05BF
```

```
; 1000          1063  1 END
; 1001          1064  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  SOR$RO_CODE                     1559  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                                Library Statistics
;
;                                          --------- Symbols ---------      Pages       Processing
;       File                               Total     Loaded    Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1         9776        148          1         581        00:01.0
;  _$255$DUA28:[SORT32.SRC]SORLIB.L32;1       409        139         33          34        00:00.4



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:SORRMSIO/OBJ=OBJ$:SORRMSIO MSRC$:SORRMSIO/UPDATE=(ENH$:SORRMSIO)

; Size:         1555 code + 4 data bytes
; Run Time:        00:37.9
; Elapsed Time:    01:54.9
; Lines/CPU Min:    1686
; Lexemes/CPU-Min: 32397
; Memory Used:  468 pages
; Compilation Complete
```

SORRMSIO
LIS

SORMSG
LIS

SORSCRIO
LIS

SOROUTPUT
LIS

SORLIB
LIS