```
   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**SORSCRIO

```
  SSSSSSSS    000000    RRRRRRRR      SSSSSSSS    CCCCCCCC  RRRRRRRR      IIIIII      000000
  SSSSSSSS    000000    RRRRRRRR      SSSSSSSS    CCCCCCCC  RRRRRRRR      IIIIII      000000
SS           00      00  RR      RR  SS          CC        RR      RR      II      00      00
SS           00      00  RR      RR  SS          CC        RR      RR      II      00      00
SS           00      00  RR      RR  SS          CC        RR      RR      II      00      00
SS           00      00  RR      RR  SS          CC        RR      RR      II      00      00
  SSSSSS     00      00  RRRRRRRR      SSSSSS    CC        RRRRRRRR        II      00      00
  SSSSSS     00      00  RRRRRRRR      SSSSSS    CC        RRRRRRRR        II      00      00
        SS   00      00  RR  RR              SS  CC        RR  RR          II      00      00
        SS   00      00  RR  RR              SS  CC        RR  RR          II      00      00
        SS   00      00  RR    RR            SS  CC        RR    RR        II      00      00      ....
        SS   00      00  RR    RR            SS  CC        RR    RR        II      00      00      ....
SSSSSSSS       000000    RR      RR  SSSSSSSS      CCCCCCCC  RR      RR    IIIIII      000000      ....
SSSSSSSS       000000    RR      RR  SSSSSSSS      CCCCCCCC  RR      RR    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
```

SO
VO

```
;     1          0001  0 MODULE SOR$SCR_IO (
;     2          0002  0                 IDENT = 'V04-000'       ! File: SORSCRIO.B32 Edit: PDG3053
;     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 RMS I/O support for work/scratch files.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  VAX/VMS user mode
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: Peter D Gilbert, CREATION DATE: 10-May-1982
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 !       T03-015         Original
;    45          0045  1 !       T03-016 Fix problem with computing COM_BUF_SIZE (due to DDB not set).
;    46          0046  1 !               Check status after calling COM_INPUT.  PDG 13-Dec-1982.
;    47          0047  1 !       T03-017 Errors opening scratch files now recoverable.  PDG 16-Dec-1982
;    48          0048  1 !       T03-018 Make write errors to a work file fatal.  PDG 22-Dec-1982
;    49          0049  1 !       T03-019 Make the size of the buffer passed to the user-written input
;    50          0050  1 !               routine for merges equal to the LRL (rather than larger).
;    51          0051  1 !               PDG 26-Dec-1982.
;    52          0052  1 !       T03-020 Changed to use QIOWs for work file I/O.
;    53          0053  1 !               Added protection XAB.  PDG 28-Dec-1982
;    54          0054  1 !       T03-021 Make work-file description blocks (WFBs) distinct from DDBs.
;    55          0055  1 !               PDG 31-Dec-1982
;    56          0056  1 !       T03-022 Added clean-up routines.  PDG 4-Jan-1983
;    57          0057  1 !       T03-023 Added a FNS parameter to WORK_OPEN.  PDG 6-Jan-1983
```

```
;    58          0058  1 ;       T03-024 Remove declaration of unreferenced externals.  PDG 26-Jan-1983
;    59          0059  1 ;       T03-025 Added WFB_USE field.  Set COM_MRG_STREAM to the file or stream
;    60          0060  1 ;               number.  PDG 27-Jan-1983
;    61          0061  1 ;       T03-026 Use COM_MERGE (rather than COM_MRG_ORDER) to indicate a merge.
;    62          0062  1 ;               PDG 31-Jan-1983
;    63          0063  1 ;       T03-027 Changes for hostile environment.  PDG 3-Feb-1983
;    64          0064  1 ;       T03-028 Pass the context address to callback routines.  PDG 11-Feb-1983
;    65          0065  1 ;       T03-029 Make SOR$_NO_WRK fatal in WORK_OPEN.  PDG 9-Mar-1983
;    66          0066  1 ;       T03-030 Maximize DEQ with TUN_K_BUFSIZE.  PDG 31-Mar-1983
;    67          0067  1 ;       T03-031 Information hiding of WFB structure.  PDG 12-Apr-1983
;    68          0068  1 ;       T03-032 Remove declaration of WFB_DEV.  PDG 13-Apr-1983
;    69          0069  1 ;       T03-033 Make DELRUN and POSITION non-global routines.  Call FLUSH with
;    70          0070  1 ;               COM_RUN_CURR in SOR$$WORK_MERGE.  Remove DST parameter from
;    71          0071  1 ;               WORK_MERGE routines.  PDG 14-Apr-1983
;    72          0072  1 ;       T03-034 Move definitions of fields specific to scratch-i/o from SORLIB
;    73          0073  1 ;               into this module.  PDG 18-Apr-1983
;    74          0074  1 ;       T03-035 Massive changes; better use of scratch file space, by allowing
;    75          0075  1 ;               runs to be composed of little pieces (RDBs).  PDG 22-Apr-1983
;    76          0076  1 ;       T03-036 Page align the work file buffers.  PDG 27-Apr-1983
;    77          0077  1 ;       T03-037 Check for more than 127 pages in WORK_WRITE.  PDG 28-Apr-1983
;    78          0078  1 ;       T03-038 Add stuff for operator assistance (in comments).  29-Apr-1983
;    79          0079  1 ;       T03-039 Free RDBs linked to RUN_BLOCKs in CLEAN_UP.
;    80          0080  1 ;               Free file name string on error in WORK_OPEN.  PDG 2-May-1983
;    81          0081  1 ;       T03-040 Change BLOCK[.NEW[RDB_WFB],WFB_FULL] to BBLOCK.  PDG 9-May-1983
;    82          0082  1 ;       T03-041 Change severity of LIB$FREE_EF errors.  PDG 6-Sep-1983
;    83          0083  1 ;               Don't FREE_VM pieces of memory gotten from GET_VM.
;    84          0084  1 ;       T03-042 Apply Jack's Law to identical pieces of code in POSITION and
;    85          0085  1 ;               SOR$$WORK_READ.
;    86          0086  1 ;       T03-043 Use the CHECKPOINT facility.  PDG 19-Sep-1983
;    87          0087  1 ;       T03-044 SOR$$BEST_FILE_NAME assumes NAM$B_RSL and NAM$B_ESL are zero
;    88          0088  1 ;               before the OPEN or CREATE.  PDG 10-Nov-1983
;    89          0089  1 ;       T03-045 Conditionalize checkpointing code.  PDG 17-Jan-1984
;    90          0090  1 ;       T03-046 Make CFT_CON_ADR relative.  PDG 25-Jan-1984
;    91          0091  1 ;       T03-046 Correct the calculation finding the address for work file names
;    92          0092  1 ;               gotten from COM_CFT_ADR.  PDG 1-Feb-1984
;    93          0093  1 ;       T03-047 Correct version ident.
;    94          0094  1 ;       T03-048 Add support for VAXELN.  Jeff East 1/2/84
;    95          0095  1 ;       T03-049 Add recovery from RMS$_RLK errors.  PDG 2-Apr-1984
;    96          0096  1 ;       T03-050 Code improvements in MRG_WORK_READ.  Clear VFC area for merges.
;    97          0097  1 ;               Improve error reporting when NO scratch files can be found.
;    98          0098  1 ;               PDG 9-Apr-1984
;    99          0099  1 ;       T03-051 Fix minor RO problem in MRG_WORK_READ for hostile environment.
;   100          0100  1 ;               PDG 15-May-1984
;   101          0101  1 ;       T03-052 Add WFB_ERRORS to avoid infinite loop when we run out of
;   102          0102  1 ;               work file space.  PDG 19-JUN-1984
;   103          0103  1 ;       T03-053 Change COM_RHB to COM_RHB_INP and COM_RHB_OUT.
;   104          0104  1 ;               This is to avoid problems with merge, where an incoming
;   105          0105  1 ;               record overwrites the VFC area for the outgoing record.
;   106          0106  1 ;               PDG 24-Jul-1984
;   107          0107  1 ;--
```

```
;  109          0108  1 LIBRARY 'SYS$LIBRARY:STARLET';
;  110          0109  1 LIBRARY 'SYS$LIBRARY:XPORT';
;  111          0110  1 REQUIRE 'SRC$:COM';
;  112          0180  1
;  113          0181  1 %IF %DECLARED(%QUOTE $DESCRIPTOR) %THEN UNDECLARE %QUOTE $DESCRIPTOR; %FI
;  114          0182  1
;  115          0183  1 FORWARD ROUTINE
;  116          0184  1     SOR$$WORK_MERGE:    CAL_CTXREG NOVALUE,      ! Determine which runs to merge
;  117          0185  1     SOR$$WORK_NEWRUN:   JSB_NEWRUN NOVALUE,      ! Start a new LONG run
;  118          0186  1     SOR$$WORK_WRITE:    JSB_OUTPUT,              ! Write to a work file
;  119          0187  1     SOR$$WORK_READ:     JSB_INPUT,               ! Read from a work file
;  120        U 0188  1 %IF FUN_K_CHECKPOINT %THEN
;  121        U 0189  1     SOR$$CHECKPOINT:    CAL_CTXREG NOVALUE,      ! Take a checkpoint
;  122        U 0190  1     PSH_CHECKPOINT,                              ! Post-checkpoint handler
;  123          0191  1 %FI
;  124          0192  1     SOR$$WRK_ALQ:       CAL_CTXREG,              ! Calculate work file allocation
;  125          0193  1     GET_BUFFER:         CAL_CTXREG NOVALUE,      ! Get a buffer
;  126          0194  1     FREE_BUFFER:        CAL_CTXREG NOVALUE,      ! Put a buffer on the free list
;  127          0195  1     POSITION:           CAL_CTXREG NOVALUE,      ! Position to start of a run
;  128          0196  1     MRG_WORK_MERGE:     CAL_CTXREG NOVALUE,      ! Set up for merge
;  129          0197  1     MRG_WORK_READ:      JSB_INPUT,               ! Read from a file for merge
;  130          0198  1     WORK_OPEN:          CAL_CTXREG NOVALUE,      ! Open a work file
;  131          0199  1     FLUSH:              CAL_CTXREG NOVALUE,      ! Flush a buffer
;  132          0200  1     GET_RDB:            CAL_CTXREG NOVALUE,      ! Get a hole
;  133          0201  1     FREE_RDB:           CAL_CTXREG NOVALUE,      ! Free a hole
;  134          0202  1     NEWRUN:             CAL_CTXREG NOVALUE,      ! Start a new run
;  135          0203  1     DELRUN:             CAL_CTXREG,              ! Indicate run no longer needed
;  136          0204  1     CLEAN_UP:           CAL_CTXREG NOVALUE;      ! Release resources
;  137          0205  1
;  138          0206  1 SOR$$END_ROUTINE_(CLEAN_UP);                     ! Declare a clean-up routine
;  139          0207  1
;  140        U 0208  1 %IF HOSTILE %THEN
;  141        U 0209  1     MACRO
;  142        U 0210  1         LIB$GET_VM = SOR$LIB$GET_VM %,
;  143        U 0211  1         SYS$QIOW = SOR$SYS$QIOW %,
;  144        U 0212  1         SYS$CREATE = SOR$SYS$CREATE %,
;  145        U 0213  1         SYS$DASSGN = SOR$SYS$DASSGN %;
;  146          0214  1 %FI
;  147          0215  1
;  148          0216  1 EXTERNAL ROUTINE
;  149          0217  1     SOR$$BEST_FILE_NAME:CAL_CTXREG NOVALUE,
;  150          0218  1     SOR$$FREE_FILE_NAME:CAL_CTXREG NOVALUE,
;  151          0219  1     SOR$$ALLOCATE:      CAL_CTXREG,                  ! Allocate storage
;  152          0220  1     SOR$$DEALLOCATE:    CAL_CTXREG NOVALUE,          ! Deallocate storage
;  153          0221  1     %IF NOT HOSTILE %THEN
;  154          0222  1     LIB$GET_EF:         ADDRESSING_MODE(GENERAL),    ! Get an event flag
;  155          0223  1     LIB$FREE_EF:        ADDRESSING_MODE(GENERAL),    ! Free event flag
;  156          0224  1     %FI
;  157          0225  1     LIB$GET_VM:         ADDRESSING_MODE(GENERAL),
;  158        U 0226  1     %IF FUN_K_CHECKPOINT %THEN
;  159        U 0227  1     CHK$CHKPNT:         ADDRESSING_MODE(GENERAL) WEAK,
;  160        U 0228  1     CHK$DCLPSH:         ADDRESSING_MODE(GENERAL) WEAK,
;  161        U 0229  1     CHK$DCLRSH:         ADDRESSING_MODE(GENERAL) WEAK,
;  162        U 0230  1     CHK$CANPSH:         ADDRESSING_MODE(GENERAL) WEAK,
;  163        U 0231  1     CHK$CANRSH:         ADDRESSING_MODE(GENERAL) WEAK,
;  164          0232  1     %FI
;  165          0233  1     SOR$$ERROR;                                  ! Issue diagnostics
```

```
;  166          0234  1
;  167        U 0235  1 %IF FUN_K_CHECKPOINT %THEN
;  168        U 0236  1 EXTERNAL LITERAL
;  169        U 0237  1     CHK$_NOTINIT:       WEAK;
;  170          0238  1 %FI
;  171          0239  1
;  172          0240  1 LITERAL
;  173          0241  1     FUN_K_ASSIST =      FALSE   AND NOT HOSTILE;
;  174          0242  1
;  175          0243  1 ! This literal determines how frequently checkpoints are done.
;  176          0244  1 ! A count-down counter, COM_COUNTDOWN is decremented by one for:
;  177          0245  1 !       each record released to / read by the sort/merge,
;  178          0246  1 !       each record returned from / written by the sort/merge, and
;  179          0247  1 !       each block written to / read from a scratch file.
;  180          0248  1 ! When the counter goes negative, a checkpoint is taken.
;  181          0249  1 !
;  182          0250  1 LITERAL
;  183          0251  1     TUN_K_ONE_MINUTE =  60000,          ! About one minute???, standalone 780
;  184          0252  1     TUN_K_COUNTDOWN =   TUN_K_ONE_MINUTE * 10;  ! Every ten minutes
;  185          0253  1
;  186          0254  1 MACRO
;  187          0255  1     FREE_RDB_(X, FLD) = FREE_RDB(X, %FIELDEXPAND(FLD,0)) %,
;  188          0256  1     MOVE_ALL_RDBS_(Q, FLD) = WHILE .Q NEQ 0 DO FREE_RDB_(Q, FLD) %;
```

```
;  190          0257  1 !                          CONTEXT AREA
;  191          0258  1 !
;  192          0259  1 !       The following fields in the context area are referenced only by this
;  193          0260  1 !       module.
;  194          0261  1 !
;  195          0262  1 $FIELD
;  196          0263  1     S_FIELDS =
;  197          0264  1         SET
;  198          0265  1         $OVERLAY(COM_SCRATCH_IO)
;  199          0266  1         S_0=            [$BYTES(0)],
;  200          0267  1         S_BUF_FREE=     [XADDR],        ! Pointer to free list of buffers
;  201          0268  1         S_BUF_SIZE=     [XLONG],        ! Size in bytes of a buffer
;  202          0269  1         S_RUN_CURR=     [XADDR],        ! Address of current run block
;  203          0270  1         S_RUN_INFO=     [XADDR],        ! Address of list of run descriptions
;  204          0271  1         S_WRK_DEQ=      [XLONG],        ! Default extend amount for work files
;  205          0272  1         S_WRK_EFN=      [XLONG],        ! Event flag to use for work files
;  206          0273  1         S_WRK_WFB=      [XADDR],        ! Address of work file WFBs
;  207          0274  1         S_BUF_ALLOC=    [XADDR],        ! Pointer to list of allocated buffers
;  208          0275  1         S_PSH_HANDLER=  [XLONG],        ! Handler-id for post-checkpoint handler
;  209          0276  1         S_RSH_HANDLER=  [XLONG],        ! Handler-id for restart handler
;  210          0277  1         S_1=            [$BYTES(0)]
;  211          0278  1         TES;
;  212          0279  1
;  213          0280  1 LITERAL TMP_K_SCRATCH = %FIELDEXPAND(S_1,0)-%FIELDEXPAND(S_0,0);
;  214          0281  1 ASSERT (TMP_K_SCRATCH LEQ COM_K_SCRATCH)
;  215        L 0282  1 %IF TMP_K_SCRATCH LSS COM_K_SCRATCH
;  216        U 0283  1 %THEN %MESSAGE('COM_K_SCRATCH can be reduced from ', %NUMBER(COM_K_SCRATCH),
;  217          0284  1         ' to ', %NUMBER(TMP_K_SCRATCH)) %FI
;  218          0285  1
;  219          0286  1 STRUCTURE
;  220          0287  1         BBLOCK[O,P,S,E;BS=0] = [BS](BBLOCK+O)<P,S,E>;
```

```
;  222          0288  1 !                          WORK FILE DESCRIPTION BLOCK
;  223          0289  1 !
;  224          0290  1 !       The WFB contains information for reading/writing a work file.
;  225          0291  1 !
;  226          0292  1 !       The media storage for each WFB is described by non-overlapping RDBs,
;  227          0293  1 !       with each media block represented in exactly one RDB.  To simplify
;  228          0294  1 !       processing, a distinguished end-of-file RDB represents a near-infinite
;  229          0295  1 !       amount of media blocks that would be available if the work file were
;  230          0296  1 !       extended.  The RDBs that are not currently holding part of a run are
;  231          0297  1 !       linked into one of three lists in the WFB.
;  232          0298  1 !
;  233          0299  1 !       WFB_FULL contains RDBs that are the least desirable for holding runs.
;  234          0300  1 !       Usually, only the end-of-file RDB is on this list; this causes other
;  235          0301  1 !       RDBs to be reused rather than increasing the work file allocation.
;  236          0302  1 !
;  237          0303  1 !       WFB_FREE and WFB_NOUSE contain RDBs that are available to hold runs.
;  238          0304  1 !       The distinction is important when checkpointing is done.  Because the
;  239          0305  1 !       information in the work files is not checkpointed, the checkpointed
;  240          0306  1 !       data structures in memory must represent a valid interpretation of the
;  241          0307  1 !       data stored in the work files, so that a restart from the previous
;  242          0308  1 !       checkpoint will still result in a correct sort.  We may overwrite data
;  243          0309  1 !       in media blocks only if the checkpointed data structures indicate that
;  244          0310  1 !       there is no useful data in those blocks.
;  245          0311  1 !
;  246          0312  1 !       WFB_NOUSE contains RDBs that have been checkpointed as containing no
;  247          0313  1 !       useful data.  WFB_FREE contains the other RDBs that contain no useful
;  248          0314  1 !       data.  Thus, RDBs should be allocated from WFB_NOUSE and released to
;  249          0315  1 !       WFB_FREE.  Just after a checkpoint is taken, the RDBs are moved from
;  250          0316  1 !       WFB_FREE to WFB_NOUSE.
;  251          0317  1 !
;  252          0318  1 $UNIT_FIELD
;  253          0319  1         WFB_FIELDS =
;  254          0320  1         SET
;  255          0321  1         WFB_NEXT=       [XADDR],                ! Pointer to next WFB
;  256          0322  1         WFB_NAME=       [$SUB_BLOCK(2)],        ! File name length/address
;  257          0323  1         WFB_ALQ=        [XLONG],                ! Allocation
;  258          0324  1         WFB_FREE=       [XADDR],                ! Free RDBs for this work file
;  259          0325  1         WFB_NOUSE=      [XADDR],                ! Free RDBs for this work file
;  260          0326  1         WFB_FULL=       [XADDR],                ! Unusable RDBs for this WFB
;  261          0327  1         WFB_ERRORS=     [XWORD],                ! Error count
;  262          0328  1         WFB_CHAN=       [XWORD]                 ! Channel number
;  263          0329  1 !       WFB_DEV=        [XLONG]                 ! Device characteristics
;  264          0330  1 !       WFB_DVI=        [$BYTES(1+NAM$S_DVI)]   ! ASCIC Device name
;  265          0331  1         TES;
;  266          0332  1 LITERAL
;  267          0333  1         WFB_K_SIZE=     $FIELD_SET_UNITS;       ! Size in bytes
;  268          0334  1 MACRO
;  269          0335  1         WFB_BLOCK=      %EXPAND $UNIT_BLOCK(WFB_K_SIZE) FIELD(WFB_FIELDS) %;
;  270          0336  1
;  271        L 0337  1 %MESSAGE('WFB_K_SIZE = ', %NUMBER(WFB_K_SIZE), ' bytes')
```

```
;  273          0338  1 !                          R U N   B L O C K
;  274          0339  1 !
;  275          0340  1 !
;  276          0341  1 !       A run is a sequence of records that are known to be in order.
;  277          0342  1 !       The merge pass merges these runs.  The information necessary to
;  278          0343  1 !       position to the start of each run is stored in the following data
;  279          0344  1 !       structure.
;  280          0345  1 !
;  281          0346  1 !       Run blocks on the S_RUN_INFO list:
;  282          0347  1 !               identify the location and length or a run of records.
;  283          0348  1 !
;  284          0349  1 !       On output:
;  285          0350  1 !               RUN_VBN   - VBN of the next block to write.
;  286          0351  1 !               RUN_SIZE  - number of blocks written to the run so far.
;  287          0352  1 !               RUN_BUFF0 - number of free bytes in the buffer.
;  288          0353  1 !               RUN_BUFF1 - address of the first free byte in the buffer.
;  289          0354  1 !
;  290          0355  1 !       On input:
;  291          0356  1 !               RUN_VBN   - VBN of the next block to read.
;  292          0357  1 !               RUN_SIZE  - number of blocks in this run.
;  293          0358  1 !               RUN_BUFF0 - bytes in the buffer that haven't been read yet.
;  294          0359  1 !               RUN_BUFF1 - address of first byte that hasn't been read yet.
;  295          0360  1 !
;  296          0361  1 $UNIT_FIELD
;  297          0362  1         RUN_FIELDS=
;  298          0363  1         SET
;  299          0364  1         RUN_NEXT=       [XADDR],                ! Pointer to next run block
;  300          0365  1         RUN_DDB=        [XADDR],                ! Addr of DDB for merge stream
;  301          0366  1         RUN_VBN=        [XLONG],                ! VBN of first block in run
;  302          0367  1         RUN_SIZE=       [XLONG],                ! Blocks in this run
;  303          0368  1         RUN_RDB_CURR=   [XADDR],                ! Current RDB
;  304          0369  1         RUN_BUFFER=     [XADDR],                ! Pointer to buffer
;  305          0370  1         RUN_BUFF0=      [XLONG],                ! Remaining length in buffer
;  306          0371  1         RUN_BUFF1=      [XADDR],                ! Address of remainder of buffer
;  307          0372  1         RUN_COUNT=      [XLONG],                ! Records in this run
;  308          0373  1         RUN_RDB=        [XADDR]                 ! List of RDBs in this run
;  309          0374  1         TES;
;  310          0375  1 LITERAL
;  311          0376  1         RUN_K_SIZE=     $FIELD_SET_UNITS;       ! Length in bytes
;  312          0377  1 MACRO
;  313          0378  1         RUN_BLOCK=      %EXPAND $UNIT_BLOCK(RUN_K_SIZE) FIELD(RUN_FIELDS) %;
;  314          0379  1
;  315        L 0380  1 %MESSAGE('RUN_K_SIZE = ', %NUMBER(RUN_K_SIZE), ' bytes')
```

```
;  317          0381  1 !                          R U N   D E S C R I P T I O N   B L O C K
;  318          0382  1 !
;  319          0383  1 !
;  320          0384  1 !       This block describes a chunk of mass storage that can be used to hold
;  321          0385  1 !       some blocks of a run.
;  322          0386  1 !
;  323          0387  1 $UNIT_FIELD
;  324          0388  1         RDB_FIELDS=
;  325          0389  1         SET
;  326          0390  1         RDB_NEXT=       [XADDR],                ! Pointer to next RDB block
;  327          0391  1         RDB_WFB=        [XADDR],                ! Pointer to WFB for RDB
;  328          0392  1         RDB_VBN=        [XLONG],                ! VBN of first block in RDB
;  329          0393  1         RDB_SIZE=       [XLONG]                 ! Blocks in this RDB
;  330          0394  1         TES;
;  331          0395  1 LITERAL
;  332          0396  1         RDB_K_SIZE=     $FIELD_SET_UNITS;       ! Length in bytes
;  333          0397  1 MACRO
;  334          0398  1         RDB_BLOCK=      %EXPAND $UNIT_BLOCK(RDB_K_SIZE) FIELD(RDB_FIELDS) %;
;  335          0399  1
;  336        L 0400  1 %MESSAGE('RDB_K_SIZE = ', %NUMBER(RDB_K_SIZE), ' bytes')
```

```
;  338          0401  1 $UNIT_FIELD
;  339          0402  1         BUF_FIELDS=
;  340          0403  1         SET
;  341          0404  1         BUF_NEXT=       [XADDR],                ! Ptr to next free buffer
;  342          0405  1         BUF_ADDR=       [XADDR]                 ! Address of this block
;  343          0406  1         TES;
;  344          0407  1 MACRO   BUF_BLOCK=      %EXPAND $UNIT_BLOCK(0) FIELD(BUF_FIELDS) %;
;  345          0408  1
;  346          0409  1 ! Macroes to insert/remove items into/from a list, at the head/tail
;  347          0410  1 !
;  348        M 0411  1 MACRO   INSLH (LIST,ITEM,NEXT) =        ! Insert at head of list
;  349        M 0412  1             BEGIN
;  350        M 0413  1             ITEM[NEXT] = .LIST;
;  351        M 0414  1             LIST = ITEM[BASE_];
;  352          0415  1             END %;
;  353        M 0416  1 MACRO   INSLT (LIST,ITEM,NEXT) =        ! Insert at tail of list
;  354        M 0417  1             BEGIN
;  355        M 0418  1             LOCAL
;  356        M 0419  1                 PPTR: REF VECTOR[1];
;  357        M 0420  1             PPTR = LIST - BBLOCK[0,NEXT];
;  358        M 0421  1             WHILE .BBLOCK[PPTR[0],NEXT] NEQ 0 DO
;  359        M 0422  1                 PPTR = .BBLOCK[PPTR[0],NEXT];
;  360        M 0423  1             BBLOCK[PPTR[0],NEXT] = ITEM[BASE_];
;  361          0424  1             END %;
;  362        M 0425  1 MACRO   REMLH (LIST,ITEM,NEXT,ALQ) =    ! Remove from head of list
;  363        M 0426  1             BEGIN
;  364        M 0427  1             %IF %NULL(ALQ)
;  365        M 0428  1             %THEN
;  366        M 0429  1                 IF (ITEM = .LIST) NEQ 0
;  367        M 0430  1                     THEN (LIST = .ITEM[NEXT]; ITEM[NEXT] = 0; TRUE)
;  368        M 0431  1                     ELSE (FALSE)
;  369        M 0432  1             %ELSE
;  370        M 0433  1                 SOR$$DEALLOCATE(ALQ, LIST, .ITEM[NEXT]);
;  371        M 0434  1             %FI
;  372          0435  1             END %;
```

```
;  374     0436  1 ROUTINE GET_BUFFER
;  375     0437  1         (
;  376     0438  1         RUNBLOCK:       REF RUN_BLOCK           ! Addr of run description block
;  377     0439  1         ):      CAL_CTXREG NOVALUE =
;  378     0440  1 !++
;  379     0441  1 !
;  380     0442  1 ! FUNCTIONAL DESCRIPTION:
;  381     0443  1 !
;  382     0444  1 !       This routine gets a buffer for the run.
;  383     0445  1 !       It first attempts to get a buffer from the free list of buffers.
;  384     0446  1 !       If this fails, it allocates more memory for the buffer.
;  385     0447  1 !
;  386     0448  1 ! FORMAL PARAMETERS:
;  387     0449  1 !
;  388     0450  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  389     0451  1 !
;  390     0452  1 ! IMPLICIT INPUTS:
;  391     0453  1 !
;  392     0454  1 !       NONE
;  393     0455  1 !
;  394     0456  1 ! IMPLICIT OUTPUTS:
;  395     0457  1 !
;  396     0458  1 !       NONE
;  397     0459  1 !
;  398     0460  1 ! ROUTINE VALUE:
;  399     0461  1 !
;  400     0462  1 !       NONE
;  401     0463  1 !
;  402     0464  1 ! SIDE EFFECTS:
;  403     0465  1 !
;  404     0466  1 !       NONE
;  405     0467  1 !--
;  406     0468  2     BEGIN
;  407     0469  2     EXTERNAL REGISTER
;  408     0470  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  409     0471  2     LOCAL
;  410     0472  2         RUN:    REF RUN_BLOCK;          ! Addr of run description block
;  411     0473  2
;  412     0474  2
;  413     0475  2     ! See whether we already have a buffer
;  414     0476  2     !
;  415     0477  2     RUN = RUNBLOCK[BASE_];
;  416     0478  2     IF .RUN[RUN_BUFFER] EQL 0
;  417     0479  2     THEN
;  418     0480  3         BEGIN
;  419     0481  3         LOCAL
;  420     0482  3             BUFPTR:     REF BUF_BLOCK;  ! Pointer to buffer
;  421     0483  3
;  422     0484  3         ! Remove the buffer from the beginning of the list.
;  423     0485  3         ! If nothing in the list, create a new buffer.
;  424     0486  3         ! Store the address of the buffer
;  425     0487  3         !
;  426     0488  4         IF NOT REMLH_(CTX[S_BUF_FREE], BUFPTR, BUF_NEXT)
;  427     0489  3         THEN
;  428     0490  4             BEGIN
;  429     0491  4             !
;  430     0492  4             ! We want the storage to be aligned on a page boundary.
```

```
;  431          0493  4          ! Get an extra page, to guarantee sufficient page-aligned pages.
;  432          0494  4          ! Instead of deallocating the unused portions, they are used to
;  433          0495  4          ! maintain a linked list of the allocated blocks, using fields
;  434          0496  4          ! BUF_NEXT and BUF_ADDR.  The field BUF_NEXT is also used in the
;  435          0497  4          ! page-aligned portion when it is put on the free list.
;  436          0498  4          !
;  437          0499  4          LOCAL
;  438          0500  4              DEL,
;  439          0501  4              P:      REF BUF_BLOCK,
;  440          0502  4              STATUS;
;  441          0503  4          STATUS = LIB$GET_VM(%REF(.CTX[S_BUF_SIZE]+COM_K_BPERPAGE),BUFPTR);
;  442          0504  4          IF NOT .STATUS
;  443          0505  4          THEN
;  444          0506  4              RETURN SOR$$ERROR(SOR$_SHR_SYERROR, 0, .STATUS);
;  445          0507  4
;  446          0508  4          ! Determine how much free space is at the beginning.
;  447          0509  4          !
;  448          0510  4          DEL = (-.BUFPTR AND (COM_K_BPERPAGE-1));
;  449          0511  4
;  450          0512  4          ! Put the memory on the S_BUF_ALLOC list
;  451          0513  4          !
;  452          0514  4          P = BUFPTR[BASE_];
;  453          0515  4          IF .DEL LSS 2*%UPVAL        ! Enough room for BUF_NEXT and BUF_ADDR?
;  454          0516  4          THEN
;  455          0517  4              P = P[BASE_] + .CTX[S_BUF_SIZE] + 2*%UPVAL;
;  456          0518  4          P[BUF_ADDR] = BUFPTR[BASE_];
;  457          0519  4          INSLH_(CTX[S_BUF_ALLOC], P, BUF_NEXT);
;  458          0520  4
;  459          0521  4          ! Adjust BUFPTR to be page aligned
;  460          0522  4          !
;  461          0523  4          BUFPTR = .BUFPTR + .DEL;
;  462          0524  3          END;
;  463          0525  3
;  464          0526  3      RUN[RUN_BUFFER] = BUFPTR[BASE_];
;  465          0527  2      END;
;  466          0528  2
;  467          0529  2  RUN[RUN_BUFF0] = .CTX[S_BUF_SIZE];
;  468          0530  2  RUN[RUN_BUFF1] = .RUN[RUN_BUFFER];
;  469          0531  2
;  470          0532  1  END;

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

                                                        .PSECT  SOR$RO_CODE_____________________2,NOWRT,  SHR,  PIC,

                                        00000000V 00000 _CLEAN_UP:
                                                        .LONG   <CLEAN_UP-_CLEAN_UP>                            ;

                                                        .EXTRN  SOR$$BEST_FILE_NAME
                                                        .EXTRN  SOR$$FREE_FILE_NAME
                                                        .EXTRN  SOR$$ALLOCATE, SOR$$DEALLOCATE
                                                        .EXTRN  LIB$GET_EF, LIB$FREE_EF
                                                        .EXTRN  LIB$GET_VM, SOR$$ERROR

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

```
                                                000C 00000 GET_BUFFER:
                                                                .WORD   Save R2,R3                      ; 0436
                                                5E      08  C2 00002          SUBL2   #8, SP
                                                52  04  AC  D0 00005          MOVL    RUNBLOCK, RUN       ; 0477
                                                    14  A2  D5 00009          TSTL    20(RUN)             ; 0478
                                                        74  12 0000C          BNEQ    5$
                                             04 AE  00CC    CB  D0 0000E      MOVL    204(CTX), BUFPTR    ; 0488
                                                        0B  13 00014          BEQL    1$
                                           00CC CB      04  BE  D0 00016      MOVL    @BUFPTR, 204(CTX)
                                                    04      BE  D4 0001C      CLRL    @BUFPTR
                                                        5C  11 0001F          BRB     4$
                                                    04  AE  9F 00021 1$:     PUSHAB  BUFPTR              ; 0503
                                      04 AE  00D0   CB 00000200 8F  C1 00024  ADDL3   #512, 208(CTX), 4(SP)
                                                    04  AE  9F 0002F         PUSHAB  4(SP)
                                         00000000G  00      02  FB 00032     CALLS   #2, LIB$GET_VM
                                                    12      50  E8 00039     BLBS    STATUS, 2$          ; 0504
                                                            50  DD 0003C     PUSHL   STATUS              ; 0506
                                                            7E  D4 0003E     CLRL    -(SP)
                                                001C11B4    8F  DD 00040     PUSHL   #1839540
                                         00000000G  00      03  FB 00046     CALLS   #3, SOR$$ERROR
                                                            04  0004D       RET
                                                50      04  AE  CE 0004E 2$:     MNEGL   BUFPTR, R0      ; 0510
                                      53        50      09      00  EF 00052  EXTZV   #0, #9, R0, DEL
                                                50      04  AE  D0 00057     MOVL    BUFPTR, P           ; 0514
                                                08          53  D1 0005B     CMPL    DEL, #8             ; 0515
                                                            0A  18 0005E     BGEQ    3$
                                                51      50  00D0    CB  C1 00060  ADDL3   208(CTX), P, R1 ; 0517
                                                50      08  A1  9E 00066     MOVAB   8(R1), P
                                                    04  A0  04  AE  D0 0006A 3$:     MOVL    BUFPTR, 4(P)    ; 0518
                                                60      00E8    CB  D0 0006F  MOVL    232(CTX), (P)       ; 0519
                                                00E8    C6      50  D0 00074  MOVL    P, 232(CTX)
                                                  04    AE      53  C0 00079  ADDL2   DEL, BUFPTR         ; 0523
                                                  14    A2  04  AE  D0 0007D 4$:     MOVL    BUFPTR, 20(RUN) ; 0526
                                                  18    A2  00D0    CB  D0 00082 5$:     MOVL    208(CTX), 24(RUN) ; 0529
                                                  1C    A2  14      A2  D0 00088  MOVL    20(RUN), 28(RUN)    ; 0530
                                                                    04  0008D   RET                         ; 0532

; Routine Size:  142 bytes,    Routine Base:  SOR$RO_CODE + 0000
```

```
;  472     0533  1 ROUTINE FREE_BUFFER
;  473     0534  1         (
;  474     0535  1         RUNBLOCK:       REF RUN_BLOCK           ! Addr of run description block
;  475     0536  1         ):      CAL_CTXREG NOVALUE =
;  476     0537  1 !++
;  477     0538  1 !
;  478     0539  1 ! FUNCTIONAL DESCRIPTION:
;  479     0540  1 !
;  480     0541  1 !       This routine puts a buffer on the free list.
;  481     0542  1 !       It also frees the unused portion of RDB connected to the RUN block.
;  482     0543  1 !
;  483     0544  1 ! FORMAL PARAMETERS:
;  484     0545  1 !
;  485     0546  1 !       RUNBLOCK        Address of run description block
;  486     0547  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  487     0548  1 !
;  488     0549  1 ! IMPLICIT INPUTS:
;  489     0550  1 !
;  490     0551  1 !       NONE
;  491     0552  1 !
;  492     0553  1 ! IMPLICIT OUTPUTS:
;  493     0554  1 !
;  494     0555  1 !       NONE
;  495     0556  1 !
;  496     0557  1 ! ROUTINE VALUE:
;  497     0558  1 !
;  498     0559  1 !       NONE
;  499     0560  1 !
;  500     0561  1 ! SIDE EFFECTS:
;  501     0562  1 !
;  502     0563  1 !       NONE
;  503     0564  1 !--
;  504     0565  2     BEGIN
;  505     0566  2     EXTERNAL REGISTER
;  506     0567  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  507     0568  2     LOCAL
;  508     0569  2         BUFPTR: REF BUF_BLOCK,
;  509     0570  2         RUN:    REF RUN_BLOCK;
;  510     0571  2
;  511     0572  2     RUN = RUNBLOCK[BASE_];
;  512     0573  2     BUFPTR = .RUN[RUN_BUFFER];
;  513     0574  2     RUN[RUN_BUFFER] = 0;            ! Not necessary, but by doing this ...
;  514     0575  2     RUN[RUN_BUFF0] = 0;             ! ... an error elsewhere ...
;  515     0576  2     RUN[RUN_BUFF1] = 0;             ! ... should cause an access violation
;  516     0577  2
;  517     0578  2     ! Insert the buffer on the list (at the head)
;  518     0579  2     !
;  519     0580  2     INSLH_(CTX[S_BUF_FREE], BUFPTR, BUF_NEXT);
;  520     0581  2
;  521     0582  2     ! Update the run's current RDB.
;  522     0583  2     ! Put the unused space at the end of the RDB on the WFB_NOUSE list.
;  523     0584  2     !
;  524     0585  3     BEGIN
;  525     0586  3     LOCAL
;  526     0587  3         RDB:    REF RDB_BLOCK;
;  527     0588  3     RDB = .RUN[RUN_RDB_CURR];
;  528     0589  3     IF RDB[BASE_] NEQ 0 THEN
```

```
;   529        0590  3       IF .RUN[RUN_VBN] - .RDB[RDB_VBN] LSSU .RDB[RDB_SIZE]
;   530        0591  3       THEN
;   531        0592  4           BEGIN
;   532        0593  4           LOCAL
;   533        0594  4               NEW:        REF RDB_BLOCK;
;   534        0595  4           NEW = SOR$$ALLOCATE(RDB_K_SIZE);
;   535        0596  4           NEW[RDB_WFB]  = .RDB[RDB_WFB];
;   536        0597  4           NEW[RDB_SIZE] = .RDB[RDB_SIZE];
;   537        0598  4           NEW[RDB_VBN]  = .RDB[RDB_VBN];
;   538        0599  4           RDB[RDB_SIZE] = .RUN[RUN_VBN]  - .RDB[RDB_VBN];
;   539        0600  4           NEW[RDB_SIZE] = .NEW[RDB_SIZE] - .RDB[RDB_SIZE];
;   540        0601  4           NEW[RDB_VBN]  = .RDB[RDB_VBN]  + .RDB[RDB_SIZE];
;   541        0602  4           FREE_RDB_(%REF(NEW[BASE_]), WFB_NOUSE);
;   542        0603  3           END;
;   543        0604  2       END;
;   544        0605  2
;   545        0606  1       END;



                                          000C 00000 FREE_BUFFER:
                                                             .WORD   Save R2,R3                      ; 0533
                    5E              04  C2 00002             SUBL2   #4, SP
                    50        04    AC  D0 00005             MOVL    RUNBLOCK, RUN                   ; 0572
                    51        14    A0  D0 00009             MOVL    20(RUN), BUFPTR                 ; 0573
                              14    A0  7C 0000D             CLRQ    20(RUN)                         ; 0574
                              1C    A0  D4 00010             CLRL    28(RUN)                         ; 0576
                    61      00CC    CB  D0 00013             MOVL    204(CTX), (BUFPTR)              ; 0580
             00CC   CB              51  D0 00018             MOVL    BUFPTR, 204(CTX)                ;
                    52        10    A0  D0 0001D             MOVL    16(RUN), RDB                    ; 0588
                                    3D  13 00021             BEQL    1$                              ; 0589
      53      08    A0        08    A2  C3 00023             SUBL3   8(RDB), 8(RUN), R3              ; 0590
              0C    A2              53  D1 00029             CMPL    R3, 12(RDB)                     ;
                                    31  1E 0002D             BGEQU   1$                              ;
                                    10  DD 0002F             PUSHL   #16                             ; 0595
        00000000G   00              01  FB 00031             CALLS   #1, SOR$$ALLOCATE               ;
              0C    A0        0C    A2  D0 00038             MOVL    12(RDB), 12(NEW)                ; 0597
              04    A0        04    A2  7D 0003D             MOVQ    4(RDB), 4(NEW)                  ; 0596
              0C    A2              53  D0 00042             MOVL    R3, 12(RDB)                     ; 0599
              0C    A0        0C    A2  C2 00046             SUBL2   12(RDB), 12(NEW)                ; 0600
 08   A0      08    A2        0C    A2  C1 0004B             ADDL3   12(RDB), 8(RDB), 8(NEW)         ; 0601
                                    14  DD 00052             PUSHL   #20                             ; 0602
              04    AE              50  D0 00054             MOVL    NEW, 4(SP)                      ;
                              04    AE  9F 00058             PUSHAB  4(SP)                           ;
            0000V   CF              02  FB 0005B             CALLS   #2, FREE_RDB                    ;
                                        04 00060 1$:         RET                                     ; 0606

; Routine Size:  97 bytes,    Routine Base:  SOR$RO_CODE + 008E
```

```
;   547        U 0607  1 %IF %SWITCHES(DEBUG) %THEN
;   548        U 0608  1 GLOBAL ROUTINE DUMP_RDBS(PARAM): CAL_CTXREG =
;   549        U 0609  1     BEGIN
;   550        U 0610  1     EXTERNAL REGISTER
;   551        U 0611  1         CTX =   COM_PRG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;   552        U 0612  1     EXTERNAL ROUTINE
;   553        U 0613  1         SOR$$OUTPUT;
;   554        U 0614  1     MACRO
;   555        U 0615  1         DESC_(A) = UPLIT(%CHARCOUNT(A), UPLIT BYTE(A)) %;
;   556        U 0616  1     LOCAL
;   557        U 0617  1         P:      REF RDB_BLOCK;
;   558        U 0618  1     P = .PARAM;
;   559        U 0619  1     WHILE .P NEQ 0 DO
;   560        U 0620  1         BEGIN
;   561        U 0621  1         SOR$$OUTPUT(DESC_(%STRING(
;   562        U 0622  1             '! !XL:  NEXT!_!XL    WFB!_!XL    VBN!_!XL   SIZE!_!XL'),
;   563        U 0623  1             P[BASE_],
;   564        U 0624  1             .P[RDB_NEXT],
;   565        U 0625  1             .P[RDB_WFB],
;   566        U 0626  1             .P[RDB_VBN],
;   567        U 0627  1             .P[RDB_SIZE]);
;   568        U 0628  1         P = .P[RDB_NEXT];
;   569        U 0629  1         END;
;   570        U 0630  1     RETURN 0;
;   571        U 0631  1     END;
;   572        U 0632  1
;   573        U 0633  1 GLOBAL ROUTINE DUMP_RUNS: CAL_CTXREG =
;   574        U 0634  1     BEGIN
;   575        U 0635  1     EXTERNAL REGISTER
;   576        U 0636  1         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;   577        U 0637  1     EXTERNAL ROUTINE
;   578        U 0638  1         SOR$$OUTPUT;
;   579        U 0639  1     MACRO
;   580        U 0640  1         DESC_(A) = UPLIT(%CHARCOUNT(A), UPLIT BYTE(A)) %;
;   581        U 0641  1     LOCAL
;   582        U 0642  1         Q:      REF WFB_BLOCK,
;   583        U 0643  1         P:      REF RUN_BLOCK;
;   584        U 0644  1     SOR$$OUTPUT(DESC_('DUMP_RUNS'));
;   585        U 0645  1     P = .CTX[S_RUN_INFO];
;   586        U 0646  1     WHILE .P NEQ 0 DO
;   587        U 0647  1         BEGIN
;   588        U 0648  1         SOR$$OUTPUT(DESC_(%STRING(
;   589        U 0649  1             '!XL:  NEXT!_!XL    DDB!_!XL    VBN!_!XL   SIZE!_!XL!/',
;   590        U 0650  1             '!_ BUFFER!_!XL  BUFF0!_!XL  BUFF1!_!XL  COUNT!_!XL!/',
;   591        U 0651  1             '!_ RDB_CURR!_!XL!/')), P[BASE_],
;   592        U 0652  1             .P[RUN_NEXT], .P[RUN_DDB], .P[RUN_VBN], .P[RUN_SIZE],
;   593        U 0653  1             .P[RUN_BUFFER], .P[RUN_BUFF0], .P[RUN_BUFF1], .P[RUN_COUNT],
;   594        U 0654  1             .P[RUN_RDB_CURR]);
;   595        U 0655  1         DUMP_RDBS(.P[RUN_RDB]);
;   596        U 0656  1         P = .P[RUN_NEXT];
;   597        U 0657  1         END;
;   598        U 0658  1     SOR$$OUTPUT(DESC_('DUMP_WFBS'));
;   599        U 0659  1     Q = .CTX[S_WRK_WFB];
;   600        U 0660  1     WHILE .Q NEQ 0 DO
;   601        U 0661  1         BEGIN
;   602        U 0662  1         SOR$$OUTPUT(DESC_(%STRING(
;   603        U 0663  1             '!XL:   ALQ!_!XL   FREE!_!XL   FULL!_!XL   CHAN!_!XW',
```

```
;  604          U 0664  1                 '!/!   NOUSE!_!XL')),
;  605          U 0665  1                 Q[BASE_],
;  606          U 0666  1                 .Q[WFB_ALQ],
;  607          U 0667  1                 .Q[WFB_FREE],
;  608          U 0668  1                 .Q[WFB_FULL],
;  609          U 0669  1                 .Q[WFB_CHAN],
;  610          U 0670  1                 .Q[WFB_NOUSE]);
;  611          U 0671  1             DUMP_RDBS(.Q[WFB_FREE]);
;  612          U 0672  1             DUMP_RDBS(.Q[WFB_FULL]);
;  613          U 0673  1             DUMP_RUNS(.Q[WFB_NOUSE]);
;  614          U 0674  1             Q = .Q[WFB_NEXT];
;  615          U 0675  1             END;
;  616          U 0676  1         RETURN 0;
;  617          U 0677  1         END;
;  618            0678  1 %FI
```

```
;  620     0679  1 ROUTINE MRG_WORK_MERGE
;  621     0680  1         (
;  622     0681  1         CNT,                                    ! Number of runs to find
;  623     0682  1         LIST:           REF VECTOR              ! Counted list of runs to merge
;  624     0683  1         ):      CAL_CTXREG NOVALUE =
;  625     0684  1 !++
;  626     0685  1 !
;  627     0686  1 ! FUNCTIONAL DESCRIPTION:
;  628     0687  1 !
;  629     0688  1 !       This routine sets up the run information for a merge process.
;  630     0689  1 !
;  631     0690  1 !       It creates run description blocks for the runs, and allocates buffer
;  632     0691  1 !       space for each run.
;  633     0692  1 !
;  634     0693  1 ! FORMAL PARAMETERS:
;  635     0694  1 !
;  636     0695  1 !       CNT             Number of runs to find
;  637     0696  1 !       LIST            Counted list of runs to merge (output parameter)
;  638     0697  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  639     0698  1 !
;  640     0699  1 !       The output run is automatically handled by the SOR$$WORK_xxx routines.
;  641     0700  1 !
;  642     0701  1 ! IMPLICIT INPUTS:
;  643     0702  1 !
;  644     0703  1 !       NONE
;  645     0704  1 !
;  646     0705  1 ! IMPLICIT OUTPUTS:
;  647     0706  1 !
;  648     0707  1 !       NONE
;  649     0708  1 !
;  650     0709  1 ! ROUTINE VALUE:
;  651     0710  1 !
;  652     0711  1 !       Status code.
;  653     0712  1 !
;  654     0713  1 ! SIDE EFFECTS:
;  655     0714  1 !
;  656     0715  1 !       NONE
;  657     0716  1 !--
;  658     0717  2     BEGIN
;  659     0718  2     EXTERNAL REGISTER
;  660     0719  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  661     0720  2     LOCAL
;  662     0721  2         LST:    REF VECTOR,
;  663     0722  2         DDB:    REF DDB_BLOCK,
;  664     0723  2         RUN:    REF RUN_BLOCK;          ! Ptr to run block
;  665     0724  2
;  666     0725  2     ! Zero the count of runs to merge, and the output run.
;  667     0726  2     !
;  668     0727  2     LST = LIST[0];
;  669     0728  2
;  670     0729  2
;  671     0730  2     ! Calculate the buffer size that will be needed.
;  672     0731  2     ! If the record interface is being used on input, this size must include
;  673     0732  2     ! both the actual record, and the internal format record.
;  674     0733  2     !
;  675     0734  2     CTX[S_BUF_SIZE] = .CTX[COM_LRL_INT];
;  676     0735  2     DDB = .CTX[COM_INP_DDB];
```

```
;  677     0736  2      IF DDB[BASE_] EQL 0
;  678     0737  2      THEN
;  679     0738  2          CTX[S_BUF_SIZE] = .CTX[S_BUF_SIZE] + .CTX[COM_LRL];
;  680     0739  2      CTX[S_BUF_SIZE] = ROUND_(.CTX[S_BUF_SIZE], COM_K_BPERPAGE);
;  681     0740  2
;  682     0741  2
;  683     0742  2      ! Set up the run description blocks
;  684     0743  2      !
;  685     0744  2      LST[0] = .CNT;
;  686     0745  2      INCR I FROM 1 TO .LST[0] DO
;  687     0746  3          BEGIN
;  688     0747  3          !
;  689     0748  3          ! Allocate another run description block, and link to the list
;  690     0749  3          !
;  691     0750  3          RUN = SOR$$ALLOCATE(RUN_K_SIZE);        ! Allocate a new run block
;  692     0751  3          INSLH_(CTX[S_RUN_INFO], RUN, RUN_NEXT); ! Link this run block into list
;  693     0752  3          !
;  694     0753  3          ! Make the run description block reference the DDB
;  695     0754  3          !       (or the stream number for user-written input routines)
;  696     0755  3          ! Put a pointer to the run description block into the list
;  697     0756  3          ! Allocate a buffer for the internal-format records
;  698     0757  3          !       (only .CTX[COM_LRL_INT] bytes needed for the buffer)
;  699     0758  3          !
;  700     0759  3          IF .CTX[COM_INP_DDB] EQL 0
;  701     0760  3          THEN
;  702     0761  4              BEGIN
;  703     0762  4              RUN[RUN_DDB] = .I;                  ! Input stream number
;  704     0763  4              END
;  705     0764  3          ELSE
;  706     0765  4              BEGIN
;  707     0766  4              RUN[RUN_DDB] = DDB[BASE_];          ! Reference the DDB
;  708     0767  4              DDB = .DDB[DDB_NEXT];               ! Advance to next DDB
;  709     0768  3              END;
;  710     0769  3          LST[.I] = RUN[BASE_];                   ! Put into the list
;  711     0770  3          GET_BUFFER(RUN[BASE_]);                 ! Get a buffer for this run
;  712     0771  3          IF DDB[BASE_] EQL 0
;  713     0772  3          THEN
;  714     0773  4              BEGIN
;  715     0774  4              !
;  716     0775  4              ! RUN_BUFFER is the address of the internal format record
;  717     0776  4              ! RUN_BUFF0/1 is the descriptor for the actual record
;  718     0777  4              !
;  719     0778  4              ! Note that RUN_BUFF0 is set to the specified LRL value, rather
;  720     0779  4              ! than the actual amount of space available.  This means that the
;  721     0780  4              ! buffer passed to the user-written input routine is exactly the
;  722     0781  4              ! size specified by the LRL.  Also, this does not affect
;  723     0782  4              ! deallocating the buffer.
;  724     0783  4              !
;  725     0784  4              RUN[RUN_BUFF0] = .CTX[COM_LRL];
;  726     0785  4              RUN[RUN_BUFF1] = .RUN[RUN_BUFF1] + .CTX[COM_LRL_INT];
;  727     0786  3              END;
;  728     0787  2          END;
;  729     0788  2
;  730     0789  2      ! The output should be counted as another run.
;  731     0790  2      !
;  732     0791  2      CTX[COM_RUNS] = .CTX[COM_RUNS] + 1;         ! One more run batted in
;  733     0792  2
```

```
;  734          0793  1     END;


                                                003C 00000 MRG_WORK_MERGE:
                                                           .WORD   Save R2,R3,R4,R5                        ; 0679
                                    55      08  AC  D0 00002          MOVL    LIST, LST                   ; 0727
                                    50    00D0  CB  9E 00006          MOVAB   208(CTX), R0                ; 0734
                                    60    0088  CB  3C 0000B          MOVZWL  136(CTX), (R0)              ;
                                    54    009C  CB  D0 00010          MOVL    156(CTX), DDB               ; 0735
                                                08  12 00015          BNEQ    1$                          ; 0736
                                    51    0084  CB  3C 00017          MOVZWL  132(CTX), R1                ; 0738
                                    60          51  C0 0001C          ADDL2   R1, (R0)                    ;
                              51          60 000001FF  8F  C1 0001F 1$: ADDL3   #511, (R0), R1             ; 0739
                              60          51 000001FF  8F  CB 00027     BICL3   #511, R1, (R0)              ;
                                    65      04  AC  D0 0002F          MOVL    CNT, (LST)                  ; 0744
                                                53  D4 00033          CLRL    I                           ; 0745
                                                47  11 00035          BRB     5$                          ;
                                                28  DD 00037 2$:       PUSHL   #40                         ; 0750
                         00000000G      00      01  FB 00039          CALLS   #1, SOR$$ALLOCATE           ;
                                        52      50  D0 00040          MOVL    R0, RUN                     ;
                                    62    00D8  CB  D0 00043          MOVL    216(CTX), (RUN)             ; 0751
                                00D8    CB      52  D0 00048          MOVL    RUN, 216(CTX)               ;
                                    009C  CB  D5 0004D          TSTL    156(CTX)                    ; 0759
                                                06  12 00051          BNEQ    3$                          ;
                                  04    A2          53  D0 00053          MOVL    I, 4(RUN)               ; 0762
                                                07  11 00057          BRB     4$                          ; 0759
                                  04    A2          54  D0 00059 3$:      MOVL    DDB, 4(RUN)             ; 0766
                                        54          64  D0 0005D          MOVL    (DDB), DDB              ; 0767
                                      6543          52  D0 00060 4$:      MOVL    RUN, (LST)[I]           ; 0769
                                                52  DD 00064          PUSHL   RUN                         ; 0770
                                FEA6    CF          01  FB 00066          CALLS   #1, GET_BUFFER          ;
                                                54  D5 0006B          TSTL    DDB                         ; 0771
                                                0F  12 0006D          BNEQ    5$                          ;
                                  18    A2    0084  CB  3C 0006F          MOVZWL  132(CTX), 24(RUN)           ; 0784
                                        50    0088  CB  3C 00075          MOVZWL  136(CTX), R0                ; 0785
                                  1C    A2          50  C0 0007A          ADDL2   R0, 28(RUN)             ;
                           B5             53          65  F3 0007E 5$:      AOBLEQ  (LST), I, 2$            ; 0745
                                            7A  AB  B6 00082          INCW    122(CTX)                    ; 0791
                                                04  00085          RET                                 ; 0793

; Routine Size:  134 bytes,    Routine Base:  SOR$RO_CODE + 00EF
```

```
;  736     0794  1 GLOBAL ROUTINE SOR$$WORK_MERGE                    ! Determine runs to merge
;  737     0795  1         (
;  738     0796  1         CNT,                                      ! Number of runs to find
;  739     0797  1         LIST:           REF VECTOR                ! Counted list of runs to merge
;  740     0798  1         ):      CAL_CTXREG NOVALUE =
;  741     0799  1 !++
;  742     0800  1 !
;  743     0801  1 ! FUNCTIONAL DESCRIPTION:
;  744     0802  1 !
;  745     0803  1 !       This routine decides which runs to merge.
;  746     0804  1 !
;  747     0805  1 ! FORMAL PARAMETERS:
;  748     0806  1 !
;  749     0807  1 !       CNT             Number of runs to find
;  750     0808  1 !       LIST            Counted list of runs to merge (output parameter)
;  751     0809  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  752     0810  1 !
;  753     0811  1 !       The output run is automatically handled by the SOR$$WORK_xxx routines.
;  754     0812  1 !
;  755     0813  1 ! IMPLICIT INPUTS:
;  756     0814  1 !
;  757     0815  1 !       NONE
;  758     0816  1 !
;  759     0817  1 ! IMPLICIT OUTPUTS:
;  760     0818  1 !
;  761     0819  1 !       NONE
;  762     0820  1 !
;  763     0821  1 ! ROUTINE VALUE:
;  764     0822  1 !
;  765     0823  1 !       Status code.
;  766     0824  1 !
;  767     0825  1 ! SIDE EFFECTS:
;  768     0826  1 !
;  769     0827  1 !       NONE
;  770     0828  1 !--
;  771     0829  2     BEGIN
;  772     0830  2     EXTERNAL REGISTER
;  773     0831  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  774     0832  2     LOCAL
;  775     0833  2         LST:    REF VECTOR,
;  776     0834  2         RUN:    REF RUN_BLOCK;          ! Ptr to run block
;  777     0835  2     MACRO
;  778     0836  2         SIZE_(A) = (BIND X = .LST[A]: RUN_BLOCK; .X[RUN_SIZE]) %,
;  779     0837  2         SWAP_(X,Y) = (LOCAL Z; Z=.X; X=.Y; Y=.Z) %;
;  780     0838  2
;  781     0839  2
;  782     0840  2     ! If we were invoked for a merge process, call a different routine
;  783     0841  2     ! to set up the run information.
;  784     0842  2     !
;  785     0843  2     IF .CTX[COM_MERGE]
;  786     0844  2     THEN
;  787     0845  3         BEGIN
;  788     0846  3         BUILTIN
;  789     0847  3             AP,
;  790     0848  3             CALLG;
;  791     0849  3         CALLG(.AP, MRG_WORK_MERGE);
;  792     0850  3         RETURN;
```

```
;  793          0851  2          END;
;  794          0852  2
;  795          0853  2
;  796          0854  2      ! If there is anything left to be written in the last run,
;  797          0855  2      ! write it out, and free the buffer.
;  798          0856  2      !
;  799          0857  2      RUN = .CTX[S_RUN_CURR];
;  800          0858  2      FLUSH(RUN[BASE_]);
;  801          0859  2      FREE_BUFFER(RUN[BASE_]);
;  802          0860  2
;  803          0861  2
;  804          0862  2      ! Zero the count of runs to merge, and the output run.
;  805          0863  2      !
;  806          0864  2      LST = LIST[0];
;  807          0865  2      LST[0] = 0;
;  808          0866  2
;  809          0867  2
;  810          0868  2      ! Start looking at active runs.
;  811          0869  2      !
;  812          0870  2      RUN = .CTX[S_RUN_INFO];
;  813          0871  2
;  814          0872  2
;  815          0873  2      ! Put the first active runs into the list of runs to be merged,
;  816          0874  2      ! while maintaining LST[1] as the the largest run in the list.
;  817          0875  2      !
;  818          0876  2      DECR I FROM .CNT-1 TO 0 DO
;  819          0877  3          BEGIN
;  820          0878  3          IF RUN[BASE_] EQL 0 THEN EXITLOOP;
;  821          0879  3          LST[0] = .LST[0] + 1;
;  822          0880  3          LST[.LST[0]] = RUN[BASE_];
;  823          0881  3          IF .RUN[RUN_SIZE] GTR SIZE_(1) THEN SWAP_(LST[.LST[0]], LST[1]);
;  824          0882  3          RUN = .RUN[RUN_NEXT];
;  825          0883  2          END;
;  826          0884  2
;  827          0885  2
;  828          0886  2      ! Look through the rest of the active runs, trying to find shorter runs.
;  829          0887  2      ! Since LST[1] is the largest run in the list, we compare with it first.
;  830          0888  2      !
;  831          0889  2      WHILE RUN[BASE_] NEQ 0 DO                          ! While more active runs
;  832          0890  3          BEGIN
;  833          0891  4          IF .RUN[RUN_SIZE] LEQ SIZE_(1)
;  834          0892  3          THEN
;  835          0893  4              BEGIN
;  836          0894  4              LST[1] = RUN[BASE_];                   ! Replace largest with this one
;  837          0895  4              !
;  838          0896  4              ! Put the next largest run into LST[1]
;  839          0897  4              !
;  840          0898  4              DECR I FROM .CNT TO 2 DO
;  841          0899  4                  IF SIZE_(.I) GTR SIZE_(1) THEN SWAP_(LST[.I], LST[1]);
;  842          0900  3              END;
;  843          0901  3          RUN = .RUN[RUN_NEXT];
;  844          0902  2          END;
;  845          0903  2
;  846          0904  2
;  847          0905  2      ! Position to each of the runs, and start reading them.
;  848          0906  2      !
;  849          0907  2      DECR I FROM .LST[0] TO 1 DO POSITION(.LST[.I]);
```

```
;  850          0908  2
;  851          0909  2
;  852          0910  2      ! Start a new run, after making sure we really need it
;  853          0911  2      !
;  854          0912  2      IF .LST[0] NEQ 0 THEN NEWRUN();
;  855          0913  2
;  856          0914  2
;  857        L 0915  2      %IF %SWITCHES(DEBUG)
;  858        U 0916  2      %THEN
;  859        U 0917  2          BEGIN
;  860        U 0918  2          EXTERNAL ROUTINE SOR$$OUTPUT;
;  861        U 0919  2          MACRO
;  862        U 0920  2              DESC_(A) = UPLIT(%CHARCOUNT(A), UPLIT BYTE(A)) %;
;  863        U 0921  2          DUMP_RUNS();
;  864        U 0922  2          SOR$$OUTPUT(DESC_('WORK_MERGE'));
;  865        U 0923  2          INCR I FROM 1 TO .LIST[0] DO
;  866        U 0924  2              SOR$$OUTPUT(DESC_('!_Run !XL'), .LIST[.I]);
;  867        U 0925  2          SOR$$OUTPUT(DESC_('!_-->_!XL'), .CTX[S_RUN_CURR]);
;  868        U 0926  2          END;
;  869          0927  2      %FI
;  870          0928  2
;  871          0929  2
;  872          0930  1      END;


                                    003C 00000          .ENTRY  SOR$$WORK_MERGE, Save R2,R3,R4,R5          ; 0794
                             5C AB  95 00002          TSTB    92(CTX)                                    ; 0843
                                06  18 00005          BGEQ    1$                                         ;
FF6E CF                         6C  FA 00007          CALLG   (AP), MRG_WORK_MERGE                       ; 0849
                                04  00000C             RET                                            ; 0845
      53                00D4 CB  D0 0000D 1$:         MOVL    212(CTX), RUN                              ; 0857
                                53  DD 00012          PUSHL   RUN                                        ; 0858
0000V CF                        01  FB 00014          CALLS   #1, FLUSH                                  ;
                                53  DD 00019          PUSHL   RUN                                        ; 0859
FEF9  CF                        01  FB 0001B          CALLS   #1, FREE_BUFFER                            ;
      54                  08 AC  D0 00020          MOVL    LIST, LST                                  ; 0864
                                64  D4 00024          CLRL    (LST)                                      ; 0865
      53                00D8 CB  D0 00026          MOVL    216(CTX), RUN                              ; 0870
      55                  04 A4  9E 0002B          MOVAB   4(LST), R5                                 ; 0881
      52                  04 AC  D0 0002F          MOVL    CNT, I                                     ;
                                25  11 00033          BRB     4$                                         ;
                                53  D5 00035 2$:       TSTL    RUN                                        ; 0878
                                24  13 00037          BEQL    5$                                         ;
                                64  D6 00039          INCL    (LST)                                      ; 0879
           50                   64  D0 0003B          MOVL    (LST), R0                                  ; 0880
         6440                   53  D0 0003E          MOVL    RUN, (LST)[R0]                             ;
           51                   65  D0 00042          MOVL    (R5), R1                                   ; 0881
   0C      A1                0C A3  D1 00045          CMPL    12(RUN), 12(R1)                            ;
                                0B  15 0004A          BLEQ    3$                                         ;
           51                6440 D0 0004C          MOVL    (LST)[R0], Z                               ;
         6440                   65  D0 00050          MOVL    (R5), (LST)[R0]                            ;
           65                   51  D0 00054          MOVL    Z, (R5)                                    ;
           53                   63  D0 00057 3$:       MOVL    (RUN), RUN                                 ; 0882
           D8                   52  F4 0005A 4$:       SOBGEQ  I, 2$                                      ; 0876
```

```
                                                53  D5 0005D 5$:    TSTL    RUN                             ; 0889
                                                38  13 0005F        BEQL    10$                             ;
                                            50  65  D0 00061        MOVL    (R5), R0                        ; 0891
                                        0C  A0  0C  A3  D1 00064    CMPL    12(RUN), 12(R0)                 ;
                                                29  14 00069        BGTR    9$                              ;
                                            65  53  D0 0006B        MOVL    RUN, (R5)                       ; 0894
                                            50  04  AC  D0 0006E    MOVL    CNT, I                          ; 0898
                                                1B  11 00072        BRB     8$                              ;
                                            52      6440  D0 00074 6$:    MOVL    (LST)[I], R2            ; 0899
                                            51      65  D0 00078        MOVL    (R5), R1                    ;
                                        0C  A1  0C  A2  D1 0007B    CMPL    12(R2), 12(R1)                  ;
                                                0B  15 00080        BLEQ    7$                              ;
                                            51      6440  D0 00082    MOVL    (LST)[I], Z                     ;
                                        6440        65  D0 00086    MOVL    (R5), (LST)[I]                  ;
                                            65      51  D0 0008A    MOVL    Z, (R5)                         ;
                                                50  D7 0008D 7$:    DECL    I                               ;
                                            02  50  D1 0008F 8$:    CMPL    I, #2                           ;
                                                E0  18 00092        BGEQ    6$                              ;
                                            53      63  D0 00094 9$:    MOVL    (RUN), RUN                  ; 0901
                                                C4  11 00097        BRB     5$                              ; 0889
                        53                  64      01  C1 00099 10$:    ADDL3   #1, (LST), I                ; 0907
                                                08  11 0009D        BRB     12$                             ;
                                                6443  DD 0009F 11$:    PUSHL   (LST)[I]                    ;
                                    0000V   CF      01  FB 000A2    CALLS   #1, POSITION                    ;
                                            F5      53  F5 000A7 12$:    SOBGTR  I, 11$                      ;
                                                64  D5 000AA        TSTL    (LST)                           ; 0912
                                                05  13 000AC        BEQL    13$                             ;
                                    0000V   CF      00  FB 000AE    CALLS   #0, NEWRUN                      ;
                                                    04 000B3 13$:    RET                                     ; 0930

; Routine Size:  180 bytes,    Routine Base:  SOR$RO_CODE + 0175
```

```
;  874          0931  1 ROUTINE FULL_RDB(STATUS):       CAL_CTXREG NOVALUE =
;  875          0932  2     BEGIN
;  876          0933  2     EXTERNAL REGISTER
;  877          0934  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  878          0935  2     LOCAL
;  879          0936  2         RUN:    REF RUN_BLOCK,
;  880          0937  2         NEW:    REF RDB_BLOCK,
;  881          0938  2         WFB:    REF WFB_BLOCK,
;  882          0939  2         RDB:    REF RDB_BLOCK;
;  883          0940  2     RUN = .CTX[S_RUN_CURR];
;  884          0941  2     RDB = .RUN[RUN_RDB_CURR];
;  885          0942  2     WFB = .RDB[RDB_WFB];
;  886          0943  2     SOR$$ERROR(SOR$_SHR_WRITEERR, 1, WFB[WFB_NAME], .STATUS);
;  887          0944  2     IF .WFB[WFB_FULL] EQL 0 THEN WFB[WFB_ERRORS] = .WFB[WFB_ERRORS] + 1;
;  888          0945  2     NEW = SOR$$ALLOCATE(RDB_K_SIZE);
;  889          0946  2     NEW[RDB_WFB]  = .RDB[RDB_WFB];
;  890          0947  2     NEW[RDB_SIZE] = .RDB[RDB_SIZE];
;  891          0948  2     NEW[RDB_VBN]  = .RDB[RDB_VBN];
;  892          0949  2     RDB[RDB_SIZE] = .RUN[RUN_VBN]  - .RDB[RDB_VBN];
;  893          0950  2     NEW[RDB_SIZE] = .NEW[RDB_SIZE] - .RDB[RDB_SIZE];
;  894          0951  2     NEW[RDB_VBN]  = .RDB[RDB_VBN]  + .RDB[RDB_SIZE];
;  895          0952  2     INSLH_(WFB[WFB_FULL], NEW, RDB_NEXT);
;  896          0953  2
;  897          0954  2     ! The first time we get errors on this file, it may be due to
;  898          0955  2     ! a failure to extend the work file.
;  899          0956  2     ! The second time may be due to a few bad RDBs, but hopefully the
;  900          0957  2     ! end-of-file RDB is okay (the user may have EXQUOTA privilege).
;  901          0958  2     ! The third time, it's clear that the end-of-file RDB is also bad.
;  902          0959  2     !
;  903          0960  2     ! If the file is that bad, throw the RDBs away.
;  904          0961  2     !
;  905          0962  2     IF .WFB[WFB_ERRORS] GEQ 3
;  906          0963  2     THEN
;  907          0964  2         REMLH_(WFB[WFB_FULL], NEW, RDB_NEXT, RDB_K_SIZE);
;  908          0965  2
;  909          0966  1     END;


                                001C 00000 FULL_RDB:
                                                    .WORD   Save R2,R3,R4                   ; 0931
                        54  00D4    CB  D0 00002            MOVL    212(CTX), RUN           ; 0940
                        52    10    A4  D0 00007            MOVL    16(RUN), RDB            ; 0941
                        53    04    A2  D0 0000B            MOVL    4(RDB), WFB             ; 0942
                              04    AC  DD 0000F            PUSHL   STATUS                  ; 0943
                              04    A3  9F 00012            PUSHAB  4(WFB)                  ;
                                    01  DD 00015            PUSHL   #1                      ;
                          001C10D2  8F  DD 00017            PUSHL   #1839314                ;
00000000G                   00        04  FB 0001D            CALLS   #4, SOR$$ERROR          ;
                                    18  A3  D5 00024            TSTL    24(WFB)             ; 0944
                                        03  12 00027            BNEQ    1$                  ;
                                    1C  A3  B6 00029            INCW    28(WFB)             ;
                                        10  DD 0002C 1$:        PUSHL   #16                 ; 0945
00000000G                   00          01  FB 0002E            CALLS   #1, SOR$$ALLOCATE   ;
                          0C    A0    0C  A2  D0 00035            MOVL    12(RDB), 12(NEW)    ; 0947
```

```
                        04   A0      04   A2  7D 0003A        MOVQ    4(RDB), 4(NEW)                          ; 0946
                    0C  A2   08   A4  08   A2  C3 0003F        SUBL3   8(RDB), 8(RUN), 12(RDB)                 ; 0949
                        0C   A0  0C   A2  C2 00046        SUBL2   12(RDB), 12(NEW)                        ; 0950
                    08  A0   08   A2  0C   A2  C1 0004B        ADDL3   12(RDB), 8(RDB), 8(NEW)                 ; 0951
                             60   18   A3  D0 00052        MOVL    24(WFB), (NEW)                          ; 0952
                        18   A3        50  D0 00056        MOVL    NEW, 24(WFB)                            ;
                             03   1C   A3  B1 0005A        CMPW    28(WFB), #3                             ; 0962
                                       0E  1F 0005E        BLSSU   2$                                      ;
                                       60  DD 00060        PUSHL   (NEW)                                   ; 0964
                                  18   A3  9F 00062        PUSHAB  24(WFB)                                 ;
                                       10  DD 00065        PUSHL   #16                                     ;
                00000000G  00          03  FB 00067        CALLS   #3, SOR$$DEALLOCATE                     ;
                                           04 0006E 2$:     RET                                             ; 0966

; Routine Size:  111 bytes,    Routine Base:  SOR$RO_CODE + 0229
```

```
;  911          0967  1 ROUTINE FILE_EXTEND
;  912          0968  1         (
;  913          0969  1         WFB:    REF WFB_BLOCK,                  ! Work file to extend
;  914          0970  1         EXSZ                                    ! Extension quantity
;  915          0971  1         ):      CAL_CTXREG NOVALUE =
;  916          0972  2     BEGIN
;  917          0973  2     EXTERNAL REGISTER
;  918          0974  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  919          0975  2     LOCAL
;  920          0976  2         FIB:    BLOCK[FIB$K_EXTDATA, BYTE],     ! File ident block
;  921          0977  2         FIBD:   VECTOR[2],                      ! File ident block descriptor
;  922          0978  2         IOSB:   VECTOR[2],                      ! Virtual addresses
;  923          0979  2         STATUS;
;  924          0980  2
;  925          0981  2     ! Set up the file identification block and its descriptor. Then issue
;  926          0982  2     ! the QIOs which do the extend and write attributes to update the end of
;  927          0983  2     ! file block.
;  928          0984  2     !
;  929          0985  2     CH$FILL (0, FIB$K_EXTDATA, FIB[BASE_]);
;  930          0986  2     FIB[FIB$W_EXCTL] = FIB$M_EXTEND OR FIB$M_ALCONB OR FIB$M_ALDEF;
;  931          0987  2     FIB[FIB$L_EXSZ] = MAXU(.EXSZ, .CTX[S_WRK_DEQ]);
;  932          0988  2
;  933          0989  2     FIBD[0] = FIB$K_EXTDATA;
;  934          0990  2     FIBD[1] = FIB;
;  935          0991  2
;  936        P 0992  2     STATUS = $QIOW(
;  937        P 0993  2         EFN=    .CTX[S_WRK_EFN],
;  938        P 0994  2         CHAN=   .WFB[WFB_CHAN],
;  939        P 0995  2         FUNC=   IO$_MODIFY,
;  940        P 0996  2         P1=     FIBD,
;  941        P 0997  2         IOSB=   IOSB
;  942          0998  2         %IF HOSTILE_ELAN %THEN , ELAN = .CTX[COM_CTXADR] %FI );
;  943          0999  2     IF .STATUS THEN STATUS = .IOSB[0];
;  944          1000  2
;  945          1001  2     IF NOT .STATUS
;  946          1002  2     THEN
;  947          1003  3         BEGIN
;  948          1004  3         !
;  949          1005  3         ! Complain and put remainder of the current RDB on FULL list
;  950          1006  3         !
;  951          1007  3         FULL_RDB(.STATUS);
;  952          1008  3         END
;  953          1009  2     ELSE
;  954          1010  3         BEGIN
;  955          1011  3         !
;  956          1012  3         ! Update the current allocation for this file
;  957          1013  3         !
;  958          1014  3         WFB[WFB_ERRORS] = 0;
;  959          1015  3         WFB[WFB_ALQ] = .FIB[FIB$L_EXSZ] + .FIB[FIB$L_EXVBN];
;  960          1016  2         END;
;  961          1017  1     END;


                                                        .EXTRN  SYS$QIOW

                                                003C 00000 FILE_EXTEND:
```

```
                                                              .WORD   Save R2,R3,R4,R5                      ; 0967
                       5E          30  C2 00002        SUBL2   #48, SP                               ;
      20       00      6E          00  2C 00005        MOVC5   #0, (SP), #0, #32, FIB                ; 0985
                             10    AE     0000A                                                      ;
                 26    AE    8A    8F  9B 0000C        MOVZBW  #138, FIB+22                          ; 0986
                       50    08    AC  D0 00011        MOVL    EXSZ, R0                              ; 0987
               00DC    CB          50  D1 00015        CMPL    R0, 220(CTX)                          ;
                                   05  1E 0001A        BGEQU   1$                                    ;
                       50  00DC    CB  D0 0001C        MOVL    220(CTX), R0                          ;
                 28    AE          50  D0 00021 1$:    MOVL    R0, FIB+24                            ;
                 08    AE          20  D0 00025        MOVL    #32, FIBD                             ; 0989
                 0C    AE    10    AE  9E 00029        MOVAB   FIB, FIBD+4                           ; 0990
                                   7E  7C 0002E        CLRQ    -(SP)                                 ; 0998
                                   7E  7C 00030        CLRQ    -(SP)                                 ;
                                   7E  D4 00032        CLRL    -(SP)                                 ;
                             1C    AE  9F 00034        PUSHAB  FIBD                                  ;
                                   7E  7C 00037        CLRQ    -(SP)                                 ;
                             20    AE  9F 00039        PUSHAB  IOSB                                  ;
                                   36  DD 0003C        PUSHL   #54                                   ;
                       52    04    AC  D0 0003E        MOVL    WFB, R2                               ;
                       7E    1E    A2  3C 00042        MOVZWL  30(R2), -(SP)                         ;
                           00E0    CB  DD 00046        PUSHL   224(CTX)                              ;
          00000000G    00          0C  FB 0004A        CALLS   #12, SYS$QIOW                         ;
                       06          50  E9 00051        BLBC    STATUS, 2$                            ; 0999
                       50          6E  D0 00054        MOVL    IOSB, STATUS                          ;
                       08          50  E8 00057        BLBS    STATUS, 3$                            ; 1001
                                   50  DD 0005A 2$:    PUSHL   STATUS                                ; 1007
               FF30    CF          01  FB 0005C        CALLS   #1, FULL_RDB                          ;
                                       04 00061        RET                                           ; 1001
                             1C    A2  B4 00062 3$:    CLRW    28(R2)                                ; 1014
         0C    A2    28    AE    2C    AE  C1 00065        ADDL3   FIB+28, FIB+24, 12(R2)                ; 1015
                                       04 0006C        RET                                           ; 1017

; Routine Size:  109 bytes,    Routine Base:  SOR$RO_CODE + 0298
```

```
;  963          1018  1 ROUTINE FLUSH                                       ! Flush the buffer for a run
;  964          1019  1         (
;  965          1020  1         RUNBLOCK:       REF RUN_BLOCK
;  966          1021  1         ):      CAL_CTXREG NOVALUE =
;  967          1022  1 !++
;  968          1023  1 !
;  969          1024  1 ! FUNCTIONAL DESCRIPTION:
;  970          1025  1 !
;  971          1026  1 !       This routine flushes a buffer.
;  972          1027  1 !       Important note: This routine does not update RUN_BUFF1 or RUN_BUFF0.
;  973          1028  1 !
;  974          1029  1 ! FORMAL PARAMETERS:
;  975          1030  1 !
;  976          1031  1 !       RUNBLOCK        Address of the run description block.
;  977          1032  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  978          1033  1 !
;  979          1034  1 ! IMPLICIT INPUTS:
;  980          1035  1 !
;  981          1036  1 !       NONE
;  982          1037  1 !
;  983          1038  1 ! IMPLICIT OUTPUTS:
;  984          1039  1 !
;  985          1040  1 !       NONE
;  986          1041  1 !
;  987          1042  1 ! ROUTINE VALUE:
;  988          1043  1 !
;  989          1044  1 !       NONE
;  990          1045  1 !
;  991          1046  1 ! SIDE EFFECTS:
;  992          1047  1 !
;  993          1048  1 !       NONE
;  994          1049  1 !--
;  995          1050  2     BEGIN
;  996          1051  2     EXTERNAL REGISTER
;  997          1052  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  998          1053  2     LOCAL
;  999          1054  2         RUN: REF RUN_BLOCK;
; 1000          1055  2     LOCAL
; 1001          1056  2         P:      VECTOR[2];
; 1002          1057  2
; 1003          1058  2     RUN = RUNBLOCK[BASE_];
; 1004          1059  2
; 1005          1060  2     ! Store the number of bytes to write, and the address of the storage
; 1006          1061  2     !
; 1007          1062  2     P[0] = ROUND (.RUN[RUN_BUFF1]-.RUN[RUN_BUFFER], COM_K_BPERPAGE);
; 1008          1063  2     P[1] = .RUN[RUN_BUFFER];
; 1009          1064  2
; 1010          1065  2     ! Write out the buffer.
; 1011          1066  2     !
; 1012          1067  2     WHILE .P[0] GTR 0 DO
; 1013          1068  3         BEGIN
; 1014          1069  3         LOCAL
; 1015          1070  3             RDB:        REF RDB_BLOCK,
; 1016          1071  3             WFB:        REF WFB_BLOCK,
; 1017          1072  3             SIZE;                           ! Number of blocks to write
; 1018          1073  3
; 1019          1074  3         ! Compute the number of block we will write
```

```
;1020     1075  3          !
;1021     1076  3          RDB = .RUN[RUN_RDB_CURR];
;1022     1077  3          WFB = .RDB[RDB_WFB];
;1023     1078  3          SIZE = MINU(
;1024     1079  3              .P[0] ^ -LN2_(COM_K_BPERPAGE),
;1025     1080  3              .RDB[RDB_VBN] + .RDB[RDB_SIZE] - .RUN[RUN_VBN],
;1026     1081  3              127);
;1027     1082  3          IF
;1028     1083  3              .SIZE EQL 0          ! Was this RDB full?
;1029     1084  3          THEN
;1030     1085  3              GET_RDB()            ! Get another RDB to use
;1031     1086  3          ELIF
;1032     1087  3              .RUN[RUN_VBN] + .SIZE GTRU .WFB[WFB_ALQ]
;1033     1088  3          THEN
;1034     1089  4              BEGIN
;1035     1090  4              !
;1036     1091  4              ! Extend the file
;1037     1092  4              !
;1038     1093  4              FILE_EXTEND(WFB[BASE_],
;1039     1094  4                  .RUN[RUN_VBN] + .SIZE - .WFB[WFB_ALQ]);
;1040     1095  4              END
;1041     1096  3          ELSE
;1042     1097  4              BEGIN
;1043     1098  4              LOCAL
;1044     1099  4                  IOSB:   VECTOR[4,WORD],
;1045     1100  4                  STATUS;
;1046     1101  4              !
;1047     1102  4              ! Actually write some stuff
;1048     1103  4              !
;1049   P 1104  4              STATUS = $QIOW(
;1050   P 1105  4                  EFN=    .CTX[S_WRK_EFN],
;1051   P 1106  4                  CHAN=   .WFB[WFB_CHAN],
;1052   P 1107  4                  FUNC=   IO$_WRITEVBLK,
;1053   P 1108  4                  IOSB=   IOSB,
;1054   P 1109  4                  ASTADR= 0,
;1055   P 1110  4                  ASTPRM= 0,
;1056   P 1111  4                  P1=     .P[1],
;1057   P 1112  4                  P2=     .SIZE ^ LN2_(COM_K_BPERPAGE),
;1058   P 1113  4                  P3=     .RUN[RUN_VBN]
;1059     1114  4                  %IF HOSTILE_ELAN %THEN , ELAN = .CTX[COM_CTXADR] %FI );
;1060     1115  4              IF .STATUS THEN STATUS = .IOSB[0];
;1061     1116  4
;1062     1117  4              IF NOT .STATUS
;1063     1118  4              THEN
;1064     1119  5                  BEGIN
;1065     1120  5                  !
;1066     1121  5                  ! Complain and put remainder of the current RDB on FULL list
;1067     1122  5                  !
;1068     1123  5                  FULL_RDB(.STATUS);
;1069     1124  5                  END
;1070     1125  4              ELSE
;1071     1126  5                  BEGIN
;1072     1127  5                  !
;1073     1128  5                  ! Update the next virtual block number to write.
;1074     1129  5                  ! Update the number of blocks in this run.
;1075     1130  5                  ! Update blocks read/written since last checkpoint.
;1076     1131  5                  ! Update the size/address of the buffer to write.
```

```
;1077          1132  5                   !
;1078          1133  5                   WFB[WFB_ERRORS] = 0;
;1079          1134  5                   RUN[RUN_VBN]  = .RUN[RUN_VBN]  + .SIZE;
;1080          1135  5                   RUN[RUN_SIZE] = .RUN[RUN_SIZE] + .SIZE;
;1081        L 1136  5 %IF FUN_K_CHECKPOINT
;1082        U 1137  5 %THEN
;1083        U 1138  5                   CTX[COM_COUNTDOWN] = .CTX[COM_COUNTDOWN] - .SIZE;
;1084          1139  5 %FI
;1085          1140  5                   P[0] = .P[0] - .SIZE ^ LN2_(COM_K_BPERPAGE);
;1086          1141  5                   P[1] = .P[1] + .SIZE ^ LN2_(COM_K_BPERPAGE);
;1087          1142  4                   END;
;1088          1143  3               END;
;1089          1144  2           END;
;1090          1145  2
;1091        L 1146  2 %IF FUN_K_CHECKPOINT
;1092        U 1147  2 %THEN
;1093        U 1148  2       IF .CTX[COM_COUNTDOWN] LSS 0
;1094        U 1149  2       THEN
;1095        U 1150  2           SOR$$CHECKPOINT();
;1096          1151  2 %FI
;1097          1152  2
;1098          1153  1       END;


                                    003C 00000 FLUSH:  .WORD   Save R2,R3,R4,R5                    ;1018
                              5E          10  C2 00002         SUBL2  #16, SP                      ;
                              52      04  AC  D0 00005         MOVL   RUNBLOCK, RUN                ;1058
                    50        1C    A2      14  A2  C3 00009         SUBL3  20(RUN), 28(RUN), R0         ;1062
                              50  01FF  C0  9E 0000F         MOVAB  511(R0), R0                  ;
              08    AE              50 000001FF  8F  CB 00014         BICL3  #511, R0, P                  ;
                          0C    AE      14  A2  D0 0001D         MOVL   20(RUN), P+4                 ;1063
                                08    AE  D5 00022 1$:       TSTL   P                            ;1067
                                      01  14 00025         BGTR   2$                           ;
                                          04 00027         RET                                 ;
                              50      10  A2  D0 00028 2$:       MOVL   16(RUN), RDB                 ;1076
                              54      04  A0  D0 0002C         MOVL   4(RDB), WFB                  ;1077
                    51        08    AE      F7  8F  78 00030         ASHL   #-9, P, R1                   ;1079
                    50        08    A0      0C  A0  C1 00036         ADDL3  12(RDB), 8(RDB), R0          ;1080
                              50      08  A2  C2 0003C         SUBL2  8(RUN), R0                   ;
                              50          51  D1 00040         CMPL   R1, R0                       ;
                                          03  1B 00043         BLEQU  3$                           ;
                              51          50  D0 00045         MOVL   R0, R1                       ;
                 0000007F     8F          51  D1 00048 3$:       CMPL   R1, #127                     ;1078
                                          04  1B 0004F         BLEQU  4$                           ;
                              51      7F  8F  9A 00051         MOVZBL #127, R1                     ;
                              53          51  D0 00055 4$:       MOVL   R1, SIZE                     ;
                                          07  12 00058         BNEQ   6$                           ;1083
                       0000V  CF              00  FB 0005A         CALLS  #0, GET_RDB                  ;1085
                                          C1  11 0005F 5$:       BRB    1$                           ;
                    50                53      08  A2  C1 00061 6$:       ADDL3  8(RUN), SIZE, R0             ;1087
                              0C    A4          50  D1 00066         CMPL   R0, 12(WFB)                  ;
                                          0E  1B 0006A         BLEQU  7$                           ;
                    7E                50      0C  A4  C3 0006C         SUBL3  12(WFB), R0, -(SP)           ;1094
                                          54  DD 00071         PUSHL  WFB                          ;1093
```

```
                    FF1B    CF              02  FB 00073          CALLS   #2, FILE_EXTEND                          ;
                                            A8  11 00078          BRB     1$                                      ; 1085
                                            7E  7C 0007A 7$:      CLRQ    -(SP)                                   ; 1114
                                            7E  D4 0007C          CLRL    -(SP)                                   ;
                                        08  A2  DD 0007E          PUSHL   8(RUN)                                  ;
                    55              53      09  78 00081          ASHL    #9, SIZE, R5                            ;
                                            55  DD 00085          PUSHL   R5                                      ;
                                        20  AE  DD 00087          PUSHL   P+4                                     ;
                                            7E  7C 0008A          CLRQ    -(SP)                                   ;
                                        20  AE  9F 0008C          PUSHAB  IOSB                                    ;
                                            30  DD 0008F          PUSHL   #48                                     ;
                                    7E      1E  A4  3C 00091      MOVZWL  30(WFB), -(SP)                          ;
                                        00E0    CB  DD 00095      PUSHL   224(CTX)                                ;
                00000000G           00          0C  FB 00099      CALLS   #12, SYS$QIOW                           ;
                                    06          50  E9 000A0      BLBC    STATUS, 8$                              ; 1115
                                    50          6E  3C 000A3      MOVZWL  IOSB, STATUS                            ;
                                    09          50  E8 000A6      BLBS    STATUS, 9$                              ; 1117
                                                50  DD 000A9 8$:  PUSHL   STATUS                                  ; 1123
                    FE74    CF              01  FB 000AB          CALLS   #1, FULL_RDB                            ;
                                            AD  11 000B0          BRB     5$                                      ; 1117
                                        1C  A4  B4 000B2 9$:      CLRW    28(WFB)                                 ; 1133
                        08  A2              53  C0 000B5          ADDL2   SIZE, 8(RUN)                            ; 1134
                        0C  A2              53  C0 000B9          ADDL2   SIZE, 12(RUN)                           ; 1135
                        08  AE              55  C2 000BD          SUBL2   R5, P                                   ; 1140
                        0C  AE              55  C0 000C1          ADDL2   R5, P+4                                 ; 1141
                                            98  11 000C5          BRB     5$                                      ; 1067
                                                04 000C7          RET                                             ; 1153

; Routine Size:  200 bytes,    Routine Base:  SOR$RO_CODE + 0305
```

```
;  1100          L 1154  1 %IF FUN_K_CHECKPOINT
;  1101          U 1155  1 %THEN
;  1102          U 1156  1 REQUIRE 'SRC$:CHKPNT';
;  1103            1157  1 %FI
```

```
;  1105          1158  1 GLOBAL ROUTINE SOR$$WORK_NEWRUN                       ! Start a new run
;  1106          1159  1         : JSB_NEWRUN NOVALUE =
;  1107          1160  1
;  1108          1161  1 !++
;  1109          1162  1 !
;  1110          1163  1 ! FUNCTIONAL DESCRIPTION:
;  1111          1164  1 !
;  1112          1165  1 !       This routine starts a new run.
;  1113          1166  1 !
;  1114          1167  1 ! FORMAL PARAMETERS:
;  1115          1168  1 !
;  1116          1169  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1117          1170  1 !
;  1118          1171  1 ! IMPLICIT INPUTS:
;  1119          1172  1 !
;  1120          1173  1 !       NONE
;  1121          1174  1 !
;  1122          1175  1 ! IMPLICIT OUTPUTS:
;  1123          1176  1 !
;  1124          1177  1 !       NONE
;  1125          1178  1 !
;  1126          1179  1 ! ROUTINE VALUE:
;  1127          1180  1 !
;  1128          1181  1 !       NONE
;  1129          1182  1 !
;  1130          1183  1 ! SIDE EFFECTS:
;  1131          1184  1 !
;  1132          1185  1 !       NONE
;  1133          1186  1 !--
;  1134          1187  2     BEGIN
;  1135          1188  2     EXTERNAL REGISTER
;  1136          1189  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1137          1190  2     LOCAL
;  1138          1191  2         RUN:    REF RUN_BLOCK,
;  1139          1192  2         STATUS;
;  1140          1193  2
;  1141          1194  2     ! See whether there is a current run being processed.
;  1142          1195  2     !
;  1143          1196  2     RUN = .CTX[S_RUN_CURR];
;  1144          1197  2     IF .RUN EQL 0                                   ! Any current run?
;  1145          1198  2     THEN
;  1146          1199  3         BEGIN
;  1147          1200  3         !
;  1148          1201  3         ! No runs yet.  Open scratch files.
;  1149          1202  3         !
;  1150          1203  3         IF .CTX[COM_WRK_FILES] EQL 0
;  1151          1204  3         THEN
;  1152          1205  4             BEGIN
;  1153          1206  4             SOR$$ERROR(SOR$_NO_WRK);                ! We can recover
;  1154          1207  4             CTX[COM_WRK_FILES] = DEF_WORK_FILES;    ! Use some work files
;  1155          1208  3             END;
;  1156          1209  3         WORK_OPEN();                                ! Open the work files
;  1157          1210  3
;  1158        L 1211  3 %IF FUN_K_CHECKPOINT
;  1159        U 1212  3 %THEN
;  1160        U 1213  3         ! Declare a post-checkpoint handler
;  1161        U 1214  3         !
```

```
U 3                 IF CHK$DCLPSH NEQ 0 AND NOT .CTX[COM_NOCHKPNT]
U 3                 THEN
U 3                     BEGIN
U 3                     STATUS = CHK$DCLPSH(PSH_CHECKPOINT, CTX[BASE_], CTX[S_PSH_HANDLER]);
U 3                     IF .STATUS THEN
U 3                     STATUS = CHK$DCLRSH(PSH_CHECKPOINT, CTX[BASE_], CTX[S_RSH_HANDLER]);
U 3                     IF NOT .STATUS
U 3                     THEN
U 3                         BEGIN
U 3                         SOR$$ERROR(SOR$_SHR_SYSERROR AND NOT STS$M_SEVERITY OR STS$K_ERROR,
U 3                             0, .STATUS);
U 3                         CTX[COM_NOCHKPNT] = TRUE;
U 3                         END;
U 3                     END;
  3 %FI
  3                 END
  2             ELSE
  3                 BEGIN
  3                 !
  3                 ! There is an outstanding run.  Flush it to the output file,
  3                 ! so we can start the new run on a block boundary.
  3                 !
  3                 FLUSH(RUN[BASE_]);
  3                 FREE_BUFFER(RUN[BASE_]);
  2                 END;
  2
  2             ! Start a new run
  2             !
  2             NEWRUN();
  2
  2
  2             ! From now on, use SOR$$WORK_WRITE to output records
  2             !
  2             CTX[COM_OUTPUT] = SOR$$WORK_WRITE;
  2
  2
  2             RETURN SS$_NORMAL;
  1             END;



                                 52  DD 00000 SOR$$WORK_NEWRUN::
                                                      PUSHL   R2                           ; 1158
               52      00D4  CB  DD 00002             MOVL    212(CTX), RUN                ; 1196
                             1D  12 00007             BNEQ    2$                           ; 1197
                         5A  AB  95 00009             TSTB    90(CTX)                      ; 1203
                             11  12 0000C             BNEQ    1$                           ;
                   001C8012  8F  DD 0000E             PUSHL   #1867794                     ; 1206
        00000000G  00        01  FB 00014             CALLS   #1, SOR$$ERROR               ;
               5A  AB        02  90 0001B             MOVB    #2, 90(CTX)                  ; 1207
            0000V  CF        00  FB 0001F 1$:         CALLS   #0, WORK_OPEN                ; 1209
                             0E  11 00024             BRB     3$                           ; 1197
                             52  DD 00026 2$:         PUSHL   RUN                          ; 123
             FF0B  CF        01  FB 00028             CALLS   #1, FLUSH                    ;
```

```
                                 52  DD 0002D          PUSHL   RUN                                     ; 1238
                          FC8D   CF  01  FB 0002F          CALLS   #1, FREE_BUFFER                         :
                          0000V  CF  00  FB 00034 3$:      CALLS   #0, NEWRON                              ; 1244
                            0C   AB  0000V CF  9E 00039    MOVAB   SOR$$WORK_WRITE, 12(CTX)                ; 1249
                                     04  BA 0003F          POPR    #^M<R2>                                 ; 1253
                                         05 00041          RSB                                             :

; Routine Size:  66 bytes,    Routine Base:  SOR$RO_CODE + 03CD
```

```
;  1202          1254  1 ROUTINE GET_RDB: CAL_CTXREG NOVALUE =
;  1203          1255  2     BEGIN
;  1204          1256  2     EXTERNAL REGISTER
;  1205          1257  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1206          1258  2     LOCAL
;  1207          1259  2         RUN:    REF RUN_BLOCK,
;  1208          1260  2         RDB:    REF RDB_BLOCK,
;  1209          1261  2         PREV:   REF VECTOR[1],          ! Ptr to previous
;  1210          1262  2         TRIES:  INITIAL(0);
;  1211          1263  2     LITERAL
;  1212          1264  2         K_TRIES = 3;
;  1213          1265  2     LABEL
;  1214          1266  2         AGAIN;
;  1215          1267  2
;  1216          1268  2     ! Find a hole that we can use.
;  1217          1269  2     !
;  1218          1270  2     WHILE TRUE DO
;  1219          1271  3 AGAIN:  BEGIN
;  1220          1272  3         LOCAL
;  1221          1273  3             MAX_SIZE:   INITIAL(%X'80000000'),  ! Smallest signed longword
;  1222          1274  3             WFB:        REF WFB_BLOCK;
;  1223          1275  3         LOCAL
;  1224          1276  3             ANY_FREE:   INITIAL(0);
;  1225          1277  3         MACRO
;  1226          1278  3             TRY_AGAIN = LEAVE AGAIN %;
;  1227          1279  3
;  1228          1280  3         ! Find the largest hole.
;  1229          1281  3         !
;  1230          1282  3         PREV = 0;
;  1231          1283  3         WFB = CTX[S_WRK_WFB] - BBLOCK[0,WFB_NEXT];
;  1232          1284  3         WHILE (WFB = .WFB[WFB_NEXT]) NEQ 0 DO           ! For all work files
;  1233          1285  4             BEGIN
;  1234          1286  4             LOCAL
;  1235          1287  4                 CURR_PREV: REF VECTOR[1];
;  1236          1288  4             ANY_FREE = .ANY_FREE OR .WFB[WFB_FREE];
;  1237          1289  4             CURR_PREV = WFB[WFB_NOUSE];
;  1238          1290  4             WHILE (RDB = .CURR_PREV[0]) NEQ 0 DO        ! For RDBs in WFB_NOUSE
;  1239          1291  5                 BEGIN
;  1240          1292  5                 IF .MAX_SIZE LSS .RDB[RDB_SIZE]
;  1241          1293  5                 THEN
;  1242          1294  6                     BEGIN
;  1243          1295  6                     MAX_SIZE = .RDB[RDB_SIZE];
;  1244          1296  6                     PREV = CURR_PREV[0];
;  1245          1297  5                     END;
;  1246          1298  5                 CURR_PREV = RDB[RDB_NEXT];
;  1247          1299  5                 END;
;  1248          1300  3             END;
;  1249          1301  3
;  1250          1302  3
;  1251          1303  3         ! Did we find something?
;  1252          1304  3         !
;  1253          1305  3         IF .MAX_SIZE GTR 0 THEN EXITLOOP;       ! Exit if no extend needed
;  1254        L 1306  3 %IF FUN_K_CHECKPOINT
;  1255        U 1307  3 %THEN
;  1256        U 1308  3         IF NOT .CTX[COM_NOCHKPNT] AND .ANY_FREE NEQ 0
;  1257        U 1309  3         THEN
;  1258        U 1310  3             BEGIN
```

```
;  1259        U 1311  3              !
;  1260        U 1312  3              ! Request a checkpoint.  This will cause the RDBs to move
;  1261        U 1313  3              ! from the WFB_FREE list to the WFB_NOUSE list.
;  1262        U 1314  3              !
;  1263        U 1315  3              SOR$$CHECKPOINT();
;  1264        U 1316  3              TRY_AGAIN;
;  1265        U 1317  3              END;
;  1266          1318  3 %FI
;  1267          1319  3          IF PREV[0] NEQ 0 THEN EXITLOOP;          ! Extend as needed
;  1268          1320  3
;  1269          1321  3
;  1270        L 1322  3 %IF FUN_K_ASSIST
;  1271        U 1323  3 %THEN
;  1272        U 1324  3          ! Request operator intervention
;  1273        U 1325  3          !
;  1274        U 1326  3          BEGIN
;  1275        U 1327  3          EXTERNAL ROUTINE
;  1276        U 1328  3              UTIL$ASSIST:        ADDRESSING_MODE(GENERAL);
;  1277        U 1329  3          GLOBAL LITERAL
;  1278        U 1330  3              UTIL$_OPERFAIL = SOR$_OPERFAIL,
;  1279        U 1331  3              UTIL$_OPREPLY  = SOR$_OPREPLY;
;  1280        U 1332  3          LOCAL
;  1281        U 1333  3              REPLY_DESC: VECTOR[2],
;  1282        U 1334  3              REPLY_BUFF: VECTOR[255,BYTE],
;  1283        U 1335  3              STATUS;
;  1284        U 1336  3          REPLY_DESC[0] = %ALLOCATION(REPLY_BUFF);
;  1285        U 1337  3          REPLY_DESC[1] = REPLY_BUFF;
;  1286        U 1338  3          STATUS = UTIL$ASSIST(REPLY_DESC[0],      ! Response
;  1287        U 1339  3              UPLIT(1,                             ! Signal message
;  1288        U 1340  3                  SOR$_EXTEND AND NOT STS$M_SEVERITY OR STS$K_ERROR),
;  1289        U 1341  3              5*60,                                ! Timeout period (in seconds)
;  1290        U 1342  3              UPLIT(1,SOR$_REQ_ALT));              ! Prompt message
;  1291        U 1343  3          IF .STATUS
;  1292        U 1344  3          THEN
;  1293        U 1345  3              BEGIN
;  1294        U 1346  3              TRIES = 0;
;  1295        U 1347  3              IF .REPLY_DESC[0] NEQ 0 THEN WORK_OPEN(REPLY_DESC[0]);
;  1296        U 1348  3              END;
;  1297        U 1349  3          END;
;  1298          1350  3 %FI
;  1299          1351  3          ! Try RDBs that had gotten errors.
;  1300          1352  3          ! Move the RDB's from WFB_FULL to WFB_FREE
;  1301          1353  3          !
;  1302          1354  3          WFB = CTX[S_WRK_WFB] - BBLOCK[0,WFB_NEXT];
;  1303          1355  3          WHILE (WFB = .WFB[WFB_NEXT]) NEQ 0 DO
;  1304          1356  3              MOVE_ALL_RDBS_( WFB[WFB_FULL], WFB_FREE );
;  1305          1357  3
;  1306          1358  3          ! Check how many times we've tried to get more space
;  1307          1359  3          !
;  1308          1360  3          IF (TRIES = .TRIES + 1) LSS K_TRIES THEN TRY_AGAIN;
;  1309          1361  3
;  1310          1362  3          ! At this point, just give up
;  1311          1363  3          !
;  1312          1364  3          SOR$$FATAL(SOR$_EXTEND);
;  1313          1365  2          END;
;  1314          1366  2
;  1315          1367  2      ! Remove the RDB from the free list
```

```
;  1316          1368  2     !
;  1317          1369  2     REMLH_(PREV[0], RDB, RDB_NEXT);
;  1318          1370  2
;  1319          1371  2     ! Link the RDB at the end of RDB blocks for the current run.
;  1320          1372  2     !
;  1321          1373  2     RUN = .CTX[S_RUN_CURR];
;  1322          1374  2     INSLT_(RUN[RUN_RDB], RDB, RDB_NEXT);        ! Insert at end of list
;  1323          1375  2
;  1324          1376  2     ! Copy relevant information into the RUN block
;  1325          1377  2     !
;  1326          1378  2     RUN[RUN_RDB_CURR] = RDB[BASE_];
;  1327          1379  2     RUN[RUN_VBN]  = .RDB[RDB_VBN];
;  1328          1380  2
;  1329          1381  1     END;


                                        00FC 00000 GET_RDB:.WORD   Save R2,R3,R4,R5,R6,R7                ; 1254
                                    57  D4 00002           CLRL    TRIES                                ; 1255
                          51 80000000   8F  D0 00004 1$:   MOVL    #-2147483648, MAX_SIZE               ; 1270
                                    55  7C 0000B           CLRQ    PREV                                 ; 1282
                               54 00E4  CB  9E 0000D           MOVAB   228(R11), R4                     ; 1283
                                    52  54  D0 00012           MOVL    R4, WFB                          ;
                                    52  62  D0 00015 2$:       MOVL    (WFB), WFB                       ; 1284
                                        1F  13 00018           BEQL    5$                               ;
                                    56  10  A2  C8 0001A       BISL2   16(WFB), ANY_FREE                ; 1288
                                    50  14  A2  9E 0001E       MOVAB   20(R2), CURR_PREV                ; 1289
                                    53      60  D0 00022 3$:   MOVL    (CURR_PREV), RDB                 ; 1290
                                        EE  13 00025           BEQL    2$                               ;
                                0C  A3      51  D1 00027       CMPL    MAX_SIZE, 12(RDB)                ; 1292
                                        07  18 0002B           BGEQ    4$                               ;
                                    51  0C  A3  D0 0002D       MOVL    12(RDB), MAX_SIZE                ; 1295
                                    55      50  D0 00031       MOVL    CURR_PREV, PREV                  ; 1296
                                    50      53  D0 00034 4$:   MOVL    RDB, CURR_PREV                   ; 1298
                                        E9  11 00037           BRB     3$                               ; 1290
                                            51  D5 00039 5$:   TSTL    MAX_SIZE                         ; 1305
                                        32  14 0003B           BGTR    9$                               ;
                                            55  D5 0003D       TSTL    PREV                             ; 1319
                                        2E  12 0003F           BNEQ    9$                               ;
                                    52      54  D0 00041       MOVL    R4, WFB                          ; 1354
                                    52      62  D0 00044 6$:   MOVL    (WFB), WFB                       ; 1355
                                        11  13 00047           BEQL    8$                               ;
                                    18  A2  D5 00049 7$:       TSTL    24(WFB)                          ; 1356
                                        F6  13 0004C           BEQL    6$                               ;
                                        10  DD 0004E           PUSHL   #16                              ;
                                    18  A2  9F 00050           PUSHAB  24(WFB)                          ;
                              0000V CF      02  FB 00053       CALLS   #2, FREE_RDB                     ;
                                        EF  11 00058           BRB     7$                               ;
                                            57  D6 0005A 8$:   INCL    TRIES                            ; 1360
                                    03      57  D1 0005C       CMPL    TRIES, #3                        ;
                                        A3  19 0005F           BLSS    1$                               ;
                                 001C80A4   8F  DD 00061       PUSHL   #1867940                         ; 1364
                        00000000G 00        01  FB 00067       CALLS   #1, SOR$$ERROR                   ;
                                            04 0006E           RET                                      ;
                                    53      65  D0 0006F 9$:   MOVL    (PREV), RDB                      ; 1369
```

```
                                        05  13 00072          BEQL    10$                             ;
                              65        63  D0 00074          MOVL    (RDB), (PREV)                   ;
                                        63  D4 00077          CLRL    (RDB)                           ;
                              50  00D4   CB  D0 00079 10$:     MOVL    212(CTX), RUN                   ; 1373
                              51    24   A0  9E 0007E          MOVAB   36(R0), PPTR                    ; 1374
                                        61  D5 00082 11$:     TSTL    (PPTR)                          ;
                                        05  13 00084          BEQL    12$                             ;
                              51        61  D0 00086          MOVL    (PPTR), PPTR                    ;
                                        F7  11 00089          BRB     11$                             ;
                              61        53  D0 0008B 12$:     MOVL    RDB, (PPTR)                     ;
                          10  A0        53  D0 0008E          MOVL    RDB, 16(RUN)                    ; 1378
                          08  A0    08  A3  D0 00092          MOVL    8(RDB), 8(RUN)                  ; 1379
                                        04  00097          RET                                     ; 1381

; Routine Size:  152 bytes,    Routine Base:  SOR$RO_CODE + 040F
```

```
;  1331          1382  1 ROUTINE FREE_RDB(
;  1332          1383  1         Q:      REF VECTOR[1],
;  1333          1384  1         OFFSET                          ! %FIELDEXPAND(WFB_FREE/NOUSE, 0)
;  1334          1385  1         ): CAL_CTXREG NOVALUE =
;  1335          1386  2     BEGIN
;  1336          1387  2     EXTERNAL REGISTER
;  1337          1388  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1338          1389  2     LOCAL
;  1339          1390  2         P:      REF RDB_BLOCK,
;  1340          1391  2         R:      REF RDB_BLOCK,
;  1341          1392  2         LOFF;
;  1342          1393  2
;  1343          1394  2     ! Get a pointer to the RDB being freed
;  1344          1395  2     !
;  1345          1396  2     P = .Q[0];
;  1346          1397  2
;  1347          1398  2     ! If we aren't checkpointing, free directly to the WFB_NOUSE list
;  1348          1399  2     !
;  1349        L 1400  2 %IF FUN_K_CHECKPOINT
;  1350        U 1401  2 %THEN
;  1351        U 1402  2     LOFF = .OFFSET;
;  1352        U 1403  2     IF .CTX[COM_NOCHKPNT]
;  1353        U 1404  2     THEN
;  1354          1405  2 %FI
;  1355          1406  2         LOFF = %FIELDEXPAND(WFB_NOUSE,0);
;  1356          1407  2
;  1357          1408  2     ! Hook the RDB onto the WFB's free list
;  1358          1409  2     !
;  1359          1410  2     R = BBLOCK[.P[RDB_WFB],.LOFF,0,%BPVAL,0] - BBLOCK[0,RDB_NEXT];
;  1360          1411  2     WHILE (R = .R[RDB_NEXT]) NEQ 0 DO
;  1361          1412  3         BEGIN
;  1362          1413  3         IF .P[RDB_VBN] + .P[RDB_SIZE] EQL .R[RDB_VBN]
;  1363          1414  3         OR .R[RDB_VBN] + .R[RDB_SIZE] EQL .P[RDB_VBN]
;  1364          1415  3         THEN
;  1365          1416  3             EXITLOOP;
;  1366          1417  2         END;
;  1367          1418  2
;  1368          1419  2     IF R[BASE_] EQL 0
;  1369          1420  2     THEN
;  1370          1421  3         BEGIN
;  1371          1422  3         !
;  1372          1423  3         ! Cannot coalesce the holes
;  1373          1424  3         ! Remove from the current list, and add to the free list
;  1374          1425  3         !
;  1375          1426  3         REMLH_(Q[0], P, RDB_NEXT);
;  1376          1427  3         INSLH_(BBLOCK[.P[RDB_WFB],.LOFF,0,%BPVAL,0], P, RDB_NEXT);
;  1377          1428  3         END
;  1378          1429  2     ELSE
;  1379          1430  3         BEGIN
;  1380          1431  3         !
;  1381          1432  3         ! Coalesce the holes, remove from current list, and deallocate.
;  1382          1433  3         !
;  1383          1434  3         R[RDB_VBN] = MINU(.R[RDB_VBN], .P[RDB_VBN]);
;  1384          1435  3         R[RDB_SIZE] = .P[RDB_SIZE] + .R[RDB_SIZE];
;  1385          1436  3         REMLH_(Q[0], P, RDB_NEXT, RDB_K_SIZE);
;  1386          1437  2         END;
;  1387          1438  2
```

```
;  1388          1439  1     END;


                                                000C 00000 FREE_RDB:
                                                           .WORD   Save R2,R3                      ;  1382
                        50        04     BC    D0 00002           MOVL    @Q, P                   ;  1396
                        53               14    D0 00006           MOVL    #20, LOFF               ;  1406
                51      53        04     A0    C1 00009           ADDL3   4(P), LOFF, R           ;  1410
                        51               61    D0 0000E 1$:       MOVL    (R), R                  ;  1411
                                         18    13 00011           BEQL    2$
                52   08 A0        0C     A0    C1 00013           ADDL3   12(P), 8(P), R2         ;  1413
                   08 A1                 52    D1 00019           CMPL    R2, 8(R)
                                         0C    13 0001D           BEQL    2$
                52   08 A1        0C     A1    C1 0001F           ADDL3   12(R), 8(R), R2         ;  1414
                   08 A0                 52    D1 00025           CMPL    R2, 8(P)
                                         E3    12 00029           BNEQ    1$
                                         51    D5 0002B 2$:       TSTL    R                       ;  1419
                                         1D    12 0002D           BNEQ    4$
                        50        04     BC    D0 0002F           MOVL    @Q, P                   ;  1426
                                         06    13 00033           BEQL    3$
                   04 BC                 60    D0 00035           MOVL    (P), @Q
                                         60    D4 00039           CLRL    (P)
                51      53        04     A0    C1 0003B 3$:       ADDL3   4(P), LOFF, R1          ;  1427
                        60               61    D0 00040           MOVL    (R1), (P)
                51      53        04     A0    C1 00043           ADDL3   4(P), LOFF, R1
                        61               50    D0 00048           MOVL    P, (R1)
                                               04 0004B           RET                             ;  1419
                        52        08     A1    D0 0004C 4$:       MOVL    8(R), R2                ;  1434
                   08 A0                 52    D1 00050           CMPL    R2, 8(P)
                                         04    1B 00054           BLEQU   5$
                        52        08     A0    D0 00056           MOVL    8(P), R2
                   08 A1                 52    D0 0005A 5$:       MOVL    R2, 8(R)
                   0C A1        0C     A0    C0 0005E           ADDL2   12(P), 12(R)            ;  1435
                                         60    DD 00063           PUSHL   (P)                     ;  1436
                                   04    AC    DD 00065           PUSHL   Q
                                         10    DD 00068           PUSHL   #16
                00000000G  00            03    FB 0006A           CALLS   #3, SOR$$DEALLOCATE
                                               04 00071           RET                             ;  1439

; Routine Size:  114 bytes,    Routine Base:  SOR$RO_CODE + 04A7
```

```
;  1390          1440  1 ROUTINE NEWRUN: CAL_CTXREG NOVALUE =              . Start a new run
;  1391          1441  1
;  1392          1442  1 !++
;  1393          1443  1 !
;  1394          1444  1 ! FUNCTIONAL DESCRIPTION:
;  1395          1445  1 !
;  1396          1446  1 !       This routine starts a new run.
;  1397          1447  1 !
;  1398          1448  1 ! FORMAL PARAMETERS:
;  1399          1449  1 !
;  1400          1450  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1401          1451  1 !
;  1402          1452  1 ! IMPLICIT INPUTS:
;  1403          1453  1 !
;  1404          1454  1 !       NONE
;  1405          1455  1 !
;  1406          1456  1 ! IMPLICIT OUTPUTS:
;  1407          1457  1 !
;  1408          1458  1 !       NONE
;  1409          1459  1 !
;  1410          1460  1 ! ROUTINE VALUE:
;  1411          1461  1 !
;  1412          1462  1 !       NONE
;  1413          1463  1 !
;  1414          1464  1 ! SIDE EFFECTS:
;  1415          1465  1 !
;  1416          1466  1 !       NONE
;  1417          1467  1 !--
;  1418          1468  2     BEGIN
;  1419          1469  2     EXTERNAL REGISTER
;  1420          1470  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1421          1471  2     LOCAL
;  1422          1472  2         RUN:    REF RUN_BLOCK;          ! Ptr to run block
;  1423          1473  2
;  1424          1474  2
;  1425          1475  2     ! Bump the number of runs
;  1426          1476  2     !
;  1427          1477  2     CTX[COM_RUNS] = .CTX[COM_RUNS] + 1;         ! One more run batted in
;  1428          1478  2
;  1429          1479  2     ! Create a run desription block for this new run.
;  1430          1480  2     ! Link the new run description block into the list.
;  1431          1481  2     ! Find a large hole to hold the merged run
;  1432          1482  2     !
;  1433          1483  2     RUN = SOR$$ALLOCATE(RUN_K_SIZE);
;  1434          1484  2     INSLH_(CTX[S_RUN_INFO], RUN, RUN_NEXT);
;  1435          1485  2     CTX[S_RUN_CURR] = RUN[BASE_];                   ! This is the current run
;  1436          1486  2     RUN[RUN_SIZE] = 0;                              ! Nothing written here yet
;  1437          1487  2     GET_RDB();
;  1438          1488  2
;  1439          1489  2     ! Get a buffer for this run
;  1440          1490  2     !
;  1441          1491  2     GET_BUFFER(RUN[BASE_]);
;  1442          1492  2
;  1443          1493  1     END;
```

```
                                        0004 00000 NEWRUN:  .WORD   Save R2                                 ; 1440
                                7A  AB  B6 00002            INCW    122(CTX)                                ; 1477
                                    28  DD 00005            PUSHL   #40                                     ; 1483
                00000000G       00      01  FB 00007        CALLS   #1, SOR$$ALLOCATE                       ;
                                52      50  D0 0000E        MOVL    R0, RUN                                 ;
                                62  00D8  CB  D0 00011      MOVL    216(CTX), (RUN)                         ; 1484
                            00D8    CB      52  D0 00016    MOVL    RUN, 216(CTX)                           ;
                            00D4    CB      52  D0 0001B    MOVL    RUN, 212(CTX)                           ; 1485
                                0C  A2  D4 00020            CLRL    12(RUN)                                 ; 1486
                            FECE    CF      00  FB 00023    CALLS   #0, GET_RDB                             ; 1487
                                            52  DD 00028    PUSHL   RUN                                     ; 1491
                            FAB8    CF      01  FB 0002A    CALLS   #1, GET_BUFFER                          ;
                                            04  0002F        RET                                             ; 1493

; Routine Size:  48 bytes,    Routine Base:  SOR$RO_CODE + 0519
```

```
;  1445          1494  1 GLOBAL ROUTINE SOR$$WORK_WRITE          ! Write to a work file
;  1446          1495  1         (
;  1447          1496  1         SRC_ADDR:       REF BLOCK       ! Address of internal format record
;  1448          1497  1         ):      JSB_OUTPUT =
;  1449          1498  1 !++
;  1450          1499  1 !
;  1451          1500  1 ! FUNCTIONAL DESCRIPTION:
;  1452          1501  1 !
;  1453          1502  1 !       This routine writes to a work file.
;  1454          1503  1 !
;  1455          1504  1 ! FORMAL PARAMETERS:
;  1456          1505  1 !
;  1457          1506  1 !       SRC_ADDR.ral.v  Address of internal format record
;  1458          1507  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1459          1508  1 !
;  1460          1509  1 ! IMPLICIT INPUTS:
;  1461          1510  1 !
;  1462          1511  1 !       NONE
;  1463          1512  1 !
;  1464          1513  1 ! IMPLICIT OUTPUTS:
;  1465          1514  1 !
;  1466          1515  1 !       NONE
;  1467          1516  1 !
;  1468          1517  1 ! ROUTINE VALUE:
;  1469          1518  1 !
;  1470          1519  1 !       The address of where the record was written.
;  1471          1520  1 !
;  1472          1521  1 ! SIDE EFFECTS:
;  1473          1522  1 !
;  1474          1523  1 !       NONE
;  1475          1524  1 !--
;  1476          1525  2     BEGIN
;  1477          1526  2     EXTERNAL REGISTER
;  1478          1527  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1479          1528  2     LOCAL
;  1480          1529  2         RUN:    REF RUN_BLOCK,  ! Current run block
;  1481          1530  2         LEN,                    ! Length of record to write
;  1482          1531  2         STATUS;                 ! Status
;  1483          1532  2
;  1484          1533  2     RUN = .CTX[S_RUN_CURR];
;  1485          1534  2
;  1486          1535  2     RUN[RUN_COUNT] = .RUN[RUN_COUNT] + 1;       ! One more record in this run
;  1487          1536  2
;  1488          1537  2
;  1489          1538  2     ! Set the length and address of the record to be written.
;  1490          1539  2     !
;  1491          1540  2     LEN = .CTX[COM_LRL_INT];
;  1492          1541  2     IF (RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] - .LEN) GEQ 0
;  1493          1542  2     THEN
;  1494          1543  3         BEGIN
;  1495          1544  3         LOCAL
;  1496          1545  3             TMP1;
;  1497          1546  3         CH$MOVE(.LEN, SRC_ADDR[BASE_], .RUN[RUN_BUFF1]);
;  1498          1547  3         TMP1 = .RUN[RUN_BUFF1];
;  1499          1548  3         RUN[RUN_BUFF1] = .RUN[RUN_BUFF1] + .LEN;
;  1500          1549  3         RETURN .TMP1;               ! Return address of the output record
;  1501          1550  3         END
```

```
;  1502          1551  2     ELSE
;  1503          1552  2         RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] + .LEN;        ! Reset
;  1504          1553  2
;  1505          1554  2
;  1506          1555  2     ! There isn't enough room in the buffer for this record
;  1507          1556  2     !
;  1508          1557  3     BEGIN
;  1509          1558  3     LOCAL
;  1510          1559  3         TMP1,           ! U div BPERPAGE * BPERPAGE
;  1511          1560  3         TMP2,           ! U mod BPERPAGE
;  1512          1561  3         BEND;           ! BUFF0 + BUFF1
;  1513          1562  3
;  1514          1563  3     !+
;  1515          1564  3     ! We want to return the address of the record, so we must have the entire
;  1516          1565  3     ! record available in the buffer.
;  1517          1566  3     !
;  1518          1567  3     ! In the equations below,
;  1519          1568  3     !   U = BUFF1-BUFFER
;  1520          1569  3     ! Write out the first part of the buffer
;  1521          1570  3     !   LEN:    ROUND_(U-BPERPAGE, BPERPAGE)
;  1522          1571  3     !   ADR:    BUFFER
;  1523          1572  3     ! Move the remaining bytes to the beginning of the buffer
;  1524          1573  3     !   SRCLEN: U     -ROUND_(U-BPERPAGE, BPERPAGE)
;  1525          1574  3     !   SRCADR: BUFFER+ROUND_(U-BPERPAGE, BPERPAGE)
;  1526          1575  3     !   DSTLEN: U     -ROUND_(U-BPERPAGE, BPERPAGE)
;  1527          1576  3     !   DSTADR: BUFFER
;  1528          1577  3     ! Move the new record into the buffer
;  1529          1578  3     !   DSTADR: BUFFER+U-ROUND_(U-BPERPAGE, BPERPAGE)
;  1530          1579  3     !-
;  1531          1580  3
;  1532          1581  3     ! Compute ROUND_(U-BPERPAGE, BPERPAGE) and
;  1533          1582  3     !   U - ROUND_(U-BPERPAGE, BPERPAGE)
;  1534          1583  3     !
;  1535          1584  3     TMP1 = .RUN[RUN_BUFF1] - .RUN[RUN_BUFFER];  ! This is momentarily U
;  1536          1585  3     TMP2 = .TMP1 AND     (COM_K_BPERPAGE-1);
;  1537          1586  3     TMP1 = .TMP1 AND NOT (COM_K_BPERPAGE-1);
;  1538          1587  3     BEND = .RUN[RUN_BUFF0] + .RUN[RUN_BUFF1];
;  1539          1588  3
;  1540          1589  3     ! Adjust BUFF0/BUFF1 to indicate a full buffer, and write it out
;  1541          1590  3     !
;  1542          1591  3     RUN[RUN_BUFF1] = .RUN[RUN_BUFFER] + .TMP1;
;  1543          1592  3     RUN[RUN_BUFF0] = 0;
;  1544          1593  3     FLUSH(RUN[BASE_]);
;  1545          1594  3
;  1546          1595  3     ! Move the remaining bytes to the beginning of the buffer
;  1547          1596  3     !
;  1548          1597  3     CH$MOVE(.TMP2, .RUN[RUN_BUFFER] + .TMP1, .RUN[RUN_BUFFER]);
;  1549          1598  3     RUN[RUN_BUFF1] = .RUN[RUN_BUFFER] + .TMP2;
;  1550          1599  3     RUN[RUN_BUFF0] = .BEND - .RUN[RUN_BUFF1];
;  1551          1600  3
;  1552          1601  3     ! Move the new record into the buffer
;  1553          1602  3     !
;  1554          1603  3     CH$MOVE(.LEN, SRC_ADDR[BASE_], .RUN[RUN_BUFF1]);
;  1555          1604  3     TMP1 = .RUN[RUN_BUFF1];
;  1556          1605  3     RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] - .LEN;
;  1557          1606  3     RUN[RUN_BUFF1] = .RUN[RUN_BUFF1] + .LEN;
;  1558          1607  3     RETURN .TMP1;
```

```
;  1559          1608  2     END;
;  1560          1609  2
;  1561          1610  1     END;


                                                5E              10  C2 00000 SOR$$WORK_WRITE::
                                                                          SUBL2   #16, SP                                         ;  1494
                                                56      00D4    CB  D0 00003          MOVL    212(CTX), RUN                    ;  1533
                                                      20        A6  D6 00008          INCL    32(RUN)                          ;  1535
                                                7E      0088    CB  3C 0000B          MOVZWL  136(CTX), LEN                    ;  1540
                                                      18        A6  9F 00010          PUSHAB  24(RUN)                          ;  1541
                                              00    BE        04  AE  C2 00013          SUBL2   LEN, @0(SP)                      ;
                                                                  11  19 00018          BLSS    1$                           ;
                                        1C    B6              6A        04  AE  28 0001A          MOVC3   LEN, (SRC_ADDR), @28(RUN)  ;  1546
                                                              50        1C  A6  D0 00020          MOVL    28(RUN), TMP1      ;  1547
                                              1C    A6        04        AE  C0 00024          ADDL2   LEN, 28(RUN)               ;  1548
                                                              6C        11  00029             BRB     2$                         ;  1549
                                              00    BE        04        AE  C0 0002B 1$:      ADDL2   LEN, @0(SP)                ;  1552
                                              08    AE        1C        A6  9E 00030          MOVAB   28(RUN), 8(SP)             ;  1584
                                        0C    AE        08        BE        14  A6  C3 00035          SUBL3   20(RUN), @8(SP), TMP1  ;
                                  10    AE        0C    AE                  09  00  EF 0003C          EXTZV   #0, #9, TMP1, TMP2     ;  1585
                                              0C    AE        01FF      8F  AA 00043          BICW2   #511, TMP1                 ;  1586
                                              14    AE        00        BE        08  BE  C1 00049          ADDL3   @8(SP), @0(SP), BEND   ;  1587
                                              08    BE        14        A6        0C  AE  C1 00050          ADDL3   TMP1, 20(RUN), @8(SP)  ;  1591
                                                                          00        BE  D4 00057          CLRL    @0(SP)             ;  1592
                                                                                    56  DD 0005A          PUSHL   RUN                ;  1593
                                                    FD5B    CF                      01  FB 0005C          CALLS   #1, FLUSH          ;
                                                  7E        14        A6        0C  AE  C1 00061          ADDL3   TMP1, 20(R6), -(SP)    ;  1597
                                            14    B6                  9E        14  AE  28 00067          MOVC3   TMP2, @(SP)+, @20(RUN)
                                            08    BE        14        A6        10  AE  C1 0006D          ADDL3   TMP2, 20(RUN), @8(SP)  ;  1598
                                            00    BE        14        AE        08  BE  C3 00074          SUBL3   @8(SP), BEND, @0(SP)   ;  1599
                                                                      56        08  BE  D0 0007B          MOVL    @8(SP), R6         ;  1603
                                                  66                  6A        04  AE  28 0007F          MOVC3   LEN, (SRC_ADDR), (R6)
                                                              0C      AE        08  BE  D0 00084          MOVL    @8(SP), TMP1       ;  1604
                                                              00      BE        04  AE  C2 00089          SUBL2   LEN, @0(SP)        ;  1605
                                                              08      BE        04  AE  C0 0008E          ADDL2   LEN, @8(SP)        ;  1606
                                                                      50        0C  AE  D0 00093          MOVL    TMP1, R0           ;  1607
                                                                      5E                18  C0 00097 2$:      ADDL2   #24, SP            ;  1610
                                                                                        05  0009A          RSB                        ;

; Routine Size:  155 bytes,    Routine Base:  SOR$RO_CODE + 0549
```

```
;  1563          1611  1 ROUTINE MRG_WORK_READ                       ! Read a record for merge
;  1564          1612  1         (
;  1565          1613  1         RUNBLOCK:       REF RUN_BLOCK       ! Address of run description block
;  1566          1614  1         ):      JSB_INPUT =
;  1567          1615  1 !++
;  1568          1616  1 !
;  1569          1617  1 ! FUNCTIONAL DESCRIPTION:
;  1570          1618  1 !
;  1571          1619  1 !       This routine reads a record for a merge, either by
;  1572          1620  1 !       using RMS to get the record, or
;  1573          1621  1 !       calling the user-written input routine.
;  1574          1622  1 !
;  1575          1623  1 ! FORMAL PARAMETERS:
;  1576          1624  1 !
;  1577          1625  1 !       RUNBLOCK        Address of run description block
;  1578          1626  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1579          1627  1 !
;  1580          1628  1 ! IMPLICIT INPUTS:
;  1581          1629  1 !
;  1582          1630  1 !       NONE
;  1583          1631  1 !
;  1584          1632  1 ! IMPLICIT OUTPUTS:
;  1585          1633  1 !
;  1586          1634  1 !       NONE
;  1587          1635  1 !
;  1588          1636  1 ! ROUTINE VALUE:
;  1589          1637  1 !
;  1590          1638  1 !       Address of record, or zero if end-of-run (end-of-file)
;  1591          1639  1 !
;  1592          1640  1 ! SIDE EFFECTS:
;  1593          1641  1 !
;  1594          1642  1 !       NONE
;  1595          1643  1 !--
;  1596          1644  2     BEGIN
;  1597          1645  2     EXTERNAL REGISTER
;  1598          1646  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1599          1647  2     BUILTIN
;  1600          1648  2         R0;
;  1601          1649  2     MACRO
;  1602          1650  2         RETURN_ = RETURN R0 = %;
;  1603          1651  2     MACRO
;  1604        M 1652  2         CH$FILL_(FILL,DN,DPTR) =
;  1605        M 1653  2             BEGIN
;  1606        M 1654  2             BUILTIN MOVC5;
;  1607        M 1655  2             MOVC5(%REF(0), .R0, %REF(FILL), %REF(DN), DPTR; R0);
;  1608          1656  2             END %;
;  1609          1657  2
;  1610          1658  2     ! Read one record from the file
;  1611          1659  2     !
;  1612          1660  2     WHILE TRUE DO
;  1613          1661  3         BEGIN
;  1614          1662  3         LOCAL
;  1615          1663  3             D:  VECTOR[2];              ! Descriptor
;  1616          1664  3
;  1617          1665  3         ! Grab a pointer to the DDB (if this is a merge with a user-written input
;  1618          1666  3         ! routine, this is really the number of the stream for him to read).
;  1619          1667  3         !
```

```
;  1620     1668  3          ! Note that this assignment is inside the loop, because there aren't
;  1621     1669  3          ! any registers that we can use (for DDB) that last across the call
;  1622     1670  3          ! to the input routine.
;  1623     1671  3          !
;  1624     1672  3          REGISTER
;  1625     1673  3              DDB:        REF DDB_BLOCK;          ! Pointer to DDB
;  1626     1674  3          DDB = .RUNBLOCK[RUN_DDB];
;  1627     1675  3
;  1628     1676  3
;  1629     1677  3          ! Clear this, since the user-input routine only stores a word.
;  1630     1678  3          !
;  1631     1679  3          D[0] = 0;
;  1632     1680  3
;  1633     1681  3          IF .CTX[COM_MRG_INPUT] NEQ 0
;  1634     1682  3          THEN
;  1635     1683  4              BEGIN
;  1636     1684  4
;  1637     1685  4              ! If there is a user-written input routine, call it.
;  1638     1686  4              !
;  1639     1687  4              IF .CTX[COM_HACK_2ARGS]
;  1640     1688  4              THEN
;  1641     1689  5                  R0 = (.CTX[COM_MRG_INPUT])
;  1642     1690  4                       (RUNBLOCK[RUN_BUFF0], %REF(.DDB), D[0])
;  1643     1691  4              ELSE
;  1644     1692  5                  R0 = (.CTX[COM_MRG_INPUT])
;  1645     1693  4                       (RUNBLOCK[RUN_BUFF0], %REF(.DDB), D[0], .CTX[COM_CTXADR]);
;  1646     1694  4              IF NOT .R0
;  1647     1695  4              THEN
;  1648     1696  5                  BEGIN
;  1649     1697  5                  IF .R0 NEQ SS$_ENDOFFILE        ! Unexpected error status?
;  1650     1698  5                  THEN
;  1651     1699  5                      R0 = SOR$$ERROR(SOR$_RTNERROR, 0, .R0);
;  1652     1700  5                  RETURN_ DELRUN(RUNBLOCK[BASE_]);        ! Return 0 to indicate EOF
;  1653     1701  4                  END;
;  1654     1702  4              D[1] = .RUNBLOCK[RUN_BUFF1];
;  1655     1703  4              CTX[COM_MRG_STREAM] = .DDB;
;  1656     1704  4              END
;  1657     1705  3          ELSE
;  1658     1706  4              BEGIN
;  1659     1707  4              %IF NOT HOSTILE %THEN                                                   ; R
;  1660     1708  4
;  1661     1709  4              ! Clear the VFC area
;  1662     1710  4              !
;  1663     1711  4              IF .CTX[COM_MINVFC] NEQ 0
;  1664     1712  4              THEN
;  1665     1713  4                  CH$FILL_(0, .CTX[COM_MAXVFC], .CTX[COM_RHB_INP]);
;  1666     1714  4
;  1667     1715  4              ! Now get the record
;  1668     1716  4              !
;  1669     1717  4              R0 = $GET(RAB=DDB[DDB_RAB+BASE_]);
;  1670     1718  4
;  1671     1719  4              ! Create a descriptor for the input record
;  1672     1720  4              !
;  1673     1721  4              D[0] = .DDB[DDB_RAB+RAB$W_RSZ];
;  1674     1722  4              D[1] = .DDB[DDB_RAB+RAB$L_RBF];
;  1675     1723  4              CTX[COM_MRG_STREAM] = .DDB[DDB_FIL];
;  1676     1724  4
```

```
;  1677        U 1725  4               %ELSE
;  1678        U 1726  4               RETURN_ SOR$$ERROR(SOR$_SHR_BADLOGIC);
;  1679          1727  4               %FI
;  1680          1728  3               END;
;  1681          1729  3
;  1682          1730  3           IF .R0
;  1683          1731  3           THEN
;  1684          1732  4               BEGIN
;  1685          1733  4
;  1686          1734  4               CTX[COM_INP_RECNUM] = .CTX[COM_INP_RECNUM] + 1;
;  1687          1735  4
;  1688          1736  4               IF
;  1689          1737  5                   BEGIN
;  1690          1738  5                   !
;  1691          1739  5                   ! Check the length
;  1692          1740  5                   !
;  1693          1741  5                   IF .D[0] GTRU .CTX[COM_LRL]
;  1694          1742  5                   THEN
;  1695          1743  6                       BEGIN
;  1696          1744  6                       !
;  1697          1745  6                       ! If too long, complain and shorten the record
;  1698          1746  6                       !
;  1699          1747  6                       R0 = SOR$$ERROR(SOR$_BAD_LRL, 1, .D[0],
;  1700          1748  6                           SOR$_SHR_READERR, 1, DDB[DDB_NAME]);
;  1701          1749  6                       D[0] = .CTX[COM_LRL];
;  1702          1750  6                       TRUE
;  1703          1751  6                       END
;  1704          1752  5                   ELIF .D[0] LSSU .CTX[COM_SRL]
;  1705          1753  5                   THEN
;  1706          1754  6                       BEGIN
;  1707          1755  6                       !
;  1708          1756  6                       ! If too short, complain and ignore the record
;  1709          1757  6                       !
;  1710          1758  6                       R0 = SOR$$ERROR(SOR$_BAD_SRL, 1, .D[0],
;  1711          1759  6                           SOR$_SHR_READERR, 1, DDB[DDB_NAME]);
;  1712          1760  6                       CTX[COM_OMI_RECNUM] = .CTX[COM_OMI_RECNUM] + 1;
;  1713          1761  6                       FALSE
;  1714          1762  6                       END
;  1715          1763  5                   ELSE
;  1716          1764  5                       TRUE
;  1717          1765  5                   END
;  1718          1766  4               THEN
;  1719          1767  5                   BEGIN
;  1720          1768  5                   !
;  1721          1769  5                   ! Convert the record to the internal format, and return the
;  1722          1770  5                   ! address of the internal format record.
;  1723          177   5                   !
;  1724          1772  6                   IF NOT (R0 = JSB_INPUT(.CTX[COM_INPUT], D[0],
;  1725          1773  6                           .RUNBLOCK[RUN_BUFFER]))
;  1726          1774  5                   THEN
;  1727          1775  5                       CTX[COM_OMI_RECNUM] = .CTX[COM_OMI_RECNUM] + 1
;  1728          1776  5                   ELSE
;  1729          1777  5                       RETURN_ .RUNBLOCK[RUN_BUFFER];
;  1730          1778  4                   END;
;  1731          1779  4               END
;  1732          1780  4           %IF NOT HOSTILE %THEN
;  1733          1781  3           ELIF
```

```
3           .RO EQL RMS$_RSA                       ! Record Stream Active
3       THEN
4           RO = $WAIT(RAB=DDB[DDB_RAB+BASE_])  ! Wait until not so active
3       ELIF
3           .RO EQL RMS$_RLK
3       THEN
4           BEGIN
4           !
4           ! Try setting the "read regardless of lock" bit.
4           ! If we had already tried that, exit with the error.
4           !
4           BUILTIN
4               TESTBITSS;
4           IF TESTBITSS(DDB[DDB_RAB+RAB$V_RRL]) THEN EXITLOOP;
4           RO = SOR$$ERROR(
4               SOR$_SHR_READERR AND NOT STS$M_SEVERITY OR STS$K_WARNING,
4               1, DDB[DDB_NAME],
4               .DDB[DDB_RAB+RAB$L_STS], .DDB[DDB_RAB+RAB$L_STV],
4               RMS$_OK_RRL);
4           END
4       %FI
3       ELSE
3           EXITLOOP;                              ! Some other error
2       END;
2
2
2   ! Check for the expected status
2   !
2   IF .RO NEQ RMS$_EOF
2   THEN
3       BEGIN
3       REGISTER
3           DDB:        REF DDB_BLOCK;            ! Pointer to DDB
3       DDB = .RUNBLOCK[RUN_DDB];
3       RO = SOR$$ERROR(SOR$_SHR_READERR, 1, DDB[DDB_NAME],
3           .DDB[DDB_RAB+RAB$L_STS], .DDB[DDB_RAB+RAB$L_STV]);
2       END;
2
2   RETURN_ DELRUN(RUNBLOCK[BASE_]);     ! Return 0
1   END;


                                                  .EXTRN  SYS$GET, SYS$WAIT

                              5A  DD 00000 MRG_WORK_READ:
                                                  PUSHL   R10                         ; 1611
                 5E           0C  C2 00002        SUBL2   #12, SP                     ;
                              59  DD 00005        PUSHL   R9                          ;
   51            6E           04  C1 00007 1$:    ADDL3   #4, RUNBLOCK, R1            ; 1674
                 56           61  D0 0000B        MOVL    (R1), DDB                   ;
                      08      AE  D4 0000E        CLRL    D                           ; 1679
                 52   60      AB  D0 00011        MOVL    96(CTX), R2                 ; 1681
                              5A  13 00015        BEQL    6$                          ;
   59            6E           18  C1 00017        ADDL3   #24, RUNBLOCK, R9           ; 1690
   11       5C   AB           05  E1 0001B        BBC     #5, 92(CTX), 2$             ; 1687
                      08      AE  9F 00020        PUSHAB  D                           ; 1690
```

```
                    08   AE             56 D0 00023       MOVL    DDB, 8(SP)                          ;
                              08        AE 9F 00027       PUSHAB  8(SP)                               ;
                                        59 DD 0002A       PUSHL   R9                                  ;
                         62             03 FB 0002C       CALLS   #3, (R2)                            ;
                                        12 11 0002F       BRB     3$                                  ; 1689
                              54        AB DD 00031 2$:   PUSHL   84(CTX)                             ; 1693
                              0C        AE 9F 00034       PUSHAB  D                                   ;
                    0C   AE             56 D0 00037       MOVL    DDB, 12(SP)                         ;
                              0C        AE 9F 0003B       PUSHAB  12(SP)                              ;
                                        59 DD 0003E       PUSHL   R9                                  ;
                         62             04 FB 00040       CALLS   #4, (R2)                            ;
                         1D             50 E8 00043 3$:   BLBS    R0, 5$                              ; 1694
              00000870   8F             50 D1 00046       CMPL    R0, #2160                           ; 1697
                                        11 13 0004D       BEQL    4$                                  ;
                                        50 DD 0004F       PUSHL   R0                                  ; 1699
                                        7E D4 00051       CLRL    -(SP)                               ;
                        001C812A        8F DD 00053       PUSHL   #1868074                            ;
              00000000G  00             03 FB 00059       CALLS   #3, SOR$$ERROR                      ;
                                   0120 31 00060 4$:      BRW     17$                                 ; 1700
               51        6E             1C C1 00063 5$:   ADDL3   #28, RUNBLOCK, R1                   ; 1702
                    0C   AE             61 D0 00067       MOVL    (R1), D+4                           ;
                    64   AB             56 D0 0006B       MOVL    DDB, 100(CTX)                       ; 1703
                                        2C 11 0006F       BRB     8$                                  ; 1681
                            0081        CB 95 00071 6$:   TSTB    129(CTX)                            ; 1711
                                        0D 13 00075       BEQL    7$                                  ;
                         51 0082        CB 9A 00077       MOVZBL  130(CTX), R1                        ; 1713
    51           00      60             00 2C 0007C       MOVC5   #0, (R0), #0, R1, @140(CTX)         ;
                            008C        DB    00081                                               ;
                              14        A6 9F 00084 7$:   PUSHAB  20(DDB)                             ; 1717
              00000000G  00             01 FB 00087       CALLS   #1, SYS$GET                         ;
                    08   AE   36        A6 3C 0008E       MOVZWL  54(DDB), D                          ; 1721
                    0C   AE   3C        A6 D0 00093       MOVL    60(DDB), D+4                        ; 1722
                    64   AB   58        A6 9A 00098       MOVZBL  88(DDB), 100(CTX)                   ; 1723
                         7B             50 E9 0009D 8$:   BLBC    R0, 13$                             ; 1730
                              7C        AB D6 000A0       INCL    124(CTX)                            ; 1734
 08   AE      0084   CB      10         00 ED 000A3       CMPZV   #0, #16, 132(CTX), D                ; 1741
                                        25 1E 000AB       BGEQU   9$                                  ;
                              04        A6 9F 000AD       PUSHAB  4(DDB)                              ; 1748
                                        01 DD 000B0       PUSHL   #1                                  ;
                        001C10B2        8F DD 000B2       PUSHL   #1839282                            ;
                              14        AE DD 000B8       PUSHL   D                                   ;
                                        01 DD 000BB       PUSHL   #1                                  ;
                        001C8082        8F DD 000BD       PUSHL   #1867906                            ;
              00000000G  00             06 FB 000C3       CALLS   #6, SOR$$ERROR                      ;
                    08   AE 0084        CB 3C 000CA       MOVZWL  132(CTX), D                         ; 1749
                                        29 11 000D0       BRB     10$                                 ;
 08   AE      0086   CB      10         00 ED 000D2 9$:   CMPZV   #0, #16, 134(CTX), D                ; 1752
                                        1F 1B 000DA       BLEQU   10$                                 ;
                              04        A6 9F 000DC       PUSHAB  4(DDB)                              ; 1759
                                        01 DD 000DF       PUSHL   #1                                  ;
                        001C10B2        8F DD 000E1       PUSHL   #1839282                            ;
                              14        AE DD 000E7       PUSHL   D                                   ;
                                        01 DD 000EA       PUSHL   #1                                  ;
                        001C80F8        8F DD 000EC       PUSHL   #1868024                            ;
              00000000G  00             06 FB 000F2       CALLS   #6, SOR$$ERROR                      ;
                                        11 11 000F9       BRB     11$                                 ; 1760
                         59   08        AE 9E 000FB 10$:  MOVAB   D, R9                               ; 1772
```

```
                7E      6E              14  C1 000FF          ADDL3   #20, RUNBLOCK, -(SP)
                        5A              9E  D0 00103          MOVL    @(SP)+, R10
                                  08    BB  16 00106          JSB     @8(CTX)
                        06              50  E8 00109          BLBS    R0, 12$
                                00BC    CB  D6 0010C 11$:     INCL    188(CTX)                    ;1775
                                        48  11 00110          BRB     15$
                51      6E              14  C1 00112 12$:     ADDL3   #20, RUNBLOCK, R1           ;1777
                        50              61  D0 00116          MOVL    (R1), R0
                                        6F  11 00119          BRB     18$                         ;1776
                000182DA 8F             50  D1 0011B 13$:     CMPL    R0, #99034                  ;1782
                                        0C  12 00122          BNEQ    14$
                                  14    A6  9F 00124          PUSHAB  20(DDB)                     ;1784
                00000000G 00            01  FB 00127          CALLS   #1, SYS$WAIT
                                        2A  11 0012E          BRB     15$
                000182AA 8F             50  D1 00130 14$:     CMPL    R0, #98986                  ;1786
                                        24  12 00137          BNEQ    16$
                1F      18      A6      03  E2 00139          BBSS    #3, 24(DDB), 16$            ;1795
                                00018029 8F  DD 0013E          PUSHL   #98345                      ;1798
                        7E        1C    A6  7D 00144          MOVQ    28(DDB), -(SP)              ;1799
                                  04    A6  9F 00148          PUSHAB  4(DDB)                      ;1798
                                        01  DD 0014B          PUSHL   #1
                                001C10B0 8F  DD 0014D          PUSHL   #1839280
                00000000G 00            06  FB 00153          CALLS   #6, SOR$$ERROR
                                      FEAA  31 0015A 15$:     BRW     1$                          ;1784
                0001827A 8F             50  D1 0015D 16$:     CMPL    R0, #98938                  ;1810
                                        1D  13 00164          BEQL    17$
                52      6E              04  C1 00166          ADDL3   #4, RUNBLOCK, R2            ;1815
                        51              62  D0 0016A          MOVL    (R2), DDB
                        7E        1C    A1  7D 0016D          MOVQ    28(DDB), -(SP)              ;1817
                                  04    A1  9F 00171          PUSHAB  4(DDB)                      ;1816
                                        01  DD 00174          PUSHL   #1
                                001C10B2 8F  DD 00176          PUSHL   #1839282
                00000000G 00            05  FB 0017C          CALLS   #5, SOR$$ERROR
                                        6E  DD 00183 17$:     PUSHL   RUNBLOCK                    ;1820
                        0000V CF        01  FB 00185          CALLS   #1, DELRUN
                              5E        10  C0 0018A 18$:     ADDL2   #16, SP                     ;1821
                              5A        8E  D0 0018D          MOVL    (SP)+, R10
                                        05  00190            RSB

; Routine Size:  401 bytes,    Routine Base:  SOR$RO_CODE + 05E4
```

```
;1775    1822  1 ROUTINE FILL                                  ! Read from a scratch file
;1776    1823  1         (
;1777    1824  1         RUNBLOCK:       REF RUN_BLOCK,  ! Address of run description block
;1778    1825  1         P:              REF VECTOR[2]   ! Length/address
;1779    1826  1         ):      JSB_INPUT =
;1780    1827  2     BEGIN
;1781    1828  2     EXTERNAL REGISTER
;1782    1829  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;1783    1830  2     LOCAL
;1784    1831  2         RUN:    REF RUN_BLOCK;
;1785    1832  2     LOCAL
;1786    1833  2         IOSB:   VECTOR[4,WORD];
;1787    1834  2
;1788    1835  2     RUN = RUNBLOCK[BASE_];
;1789    1836  2
;1790    1837  2     WHILE .P[0] GTR 0 DO
;1791    1838  3         BEGIN
;1792    1839  3         LOCAL
;1793    1840  3             RDB:        REF RDB_BLOCK,
;1794    1841  3             WFB:        REF WFB_BLOCK,
;1795    1842  3             SIZE;                           ! Number of blocks to write
;1796    1843  3
;1797    1844  3         ! Compute the number of blocks we will read
;1798    1845  3         !
;1799    1846  3         RDB = .RUN[RUN_RDB_CURR];
;1800    1847  3         WFB = .RDB[RDB_WFB];
;1801    1848  3         SIZE = MINU(
;1802    1849  3             .P[0] ^ -LN2_(COM_K_BPERPAGE),
;1803    1850  3             .RDB[RDB_VBN] + .RDB[RDB_SIZE] - .RUN[RUN_VBN],
;1804    1851  3             127);
;1805    1852  3         IF
;1806    1853  3             .SIZE EQL 0         ! Was this RDB depleted?
;1807    1854  3         THEN
;1808    1855  4             BEGIN
;1809    1856  4             FREE_RDB_(RUN[RUN_RDB], WFB_FREE);
;1810    1857  4             RUN[RUN_RDB_CURR] = .RUN[RUN_RDB];
;1811    1858  4             RDB = .RUN[RUN_RDB_CURR];
;1812    1859  4             IF RDB[BASE_] EQL 0 THEN EXITLOOP;
;1813    1860  4             RUN[RUN_VBN] = .RDB[RDB_VBN];
;1814    1861  4             END
;1815    1862  3         ELSE
;1816    1863  4             BEGIN
;1817    1864  4             LOCAL
;1818    1865  4                 STATUS;
;1819    1866  4
;1820  P 1867  4             STATUS = $QIOW(
;1821  P 1868  4                 EFN=    .CTX[S_WRK_EFN],
;1822  P 1869  4                 CHAN=   .WFB[WFB_CHAN],
;1823  P 1870  4                 FUNC=   IO$_READVBLK,
;1824  P 1871  4                 IOSB=   IOSB,
;1825  P 1872  4                 ASTADR= 0,
;1826  P 1873  4                 ASTPRM= 0,
;1827  P 1874  4                 P1=     .P[1],
;1828  P 1875  4                 P2=     .SIZE ^ LN2_(COM_K_BPERPAGE),
;1829  P 1876  4                 P3=     .RUN[RUN_VBN]
;1830    1877  4                 %IF HOSTILE_ELAN %THEN , ELAN = .CTX[COM_CTXADR] %FI );
;1831    1878  4             IF .STATUS THEN STATUS = .IOSB[0];
```

```
;                1879  4
;                1880  4                IF NOT .STATUS
;                1881  4                THEN
;                1882  5                    BEGIN
;                1883  5                    RETURN SOR$$ERROR(SOR$_SHR_READERR, 1, WFB[WFB_NAME], .STATUS);
;                1884  5                    END
;                1885  4                ELSE
;                1886  5                    BEGIN
;                1887  5                    !
;                1888  5                    ! Update the next virtual block number to read.
;                1889  5                    ! Update blocks read/written since last checkpoint.
;                1890  5                    ! Update the size/address of the buffer to write
;                1891  5                    !
;                1892  5                    RUN[RUN_VBN]  = .RUN[RUN_VBN]  + .SIZE;
;             L  1893  5 %IF FUN_K_CHECKPOINT
;             U  1894  5 %THEN
;             U  1895  5                    CTX[COM_COUNTDOWN] = .CTX[COM_COUNTDOWN] - .SIZE;
;                1896  5 %FI
;                1897  5                    P[0] = .P[0] - .SIZE ^ LN2_(COM_K_BPERPAGE);
;                1898  5                    P[1] = .P[1] + .SIZE ^ LN2_(COM_K_BPERPAGE);
;                1899  4                    END;
;                1900  3                END;
;                1901  2            END;
;                1902  2
;             L  1903  2 %IF FUN_K_CHECKPOINT
;             U  1904  2 %THEN
;             U  1905  2     IF .CTX[COM_COUNTDOWN] LSS 0
;             U  1906  2     THEN
;             U  1907  2         SOR$$CHECKPOINT();
;                1908  2 %FI
;                1909  2
;                1910  2     RETURN SS$_NORMAL;
;                1911  1     END;


                                        5A  DD 00000 FILL:   PUSHL   R10                              ; 1822
                                     5E     08  C2 00002         SUBL2   #8, SP
                                     54     59  7D 00005         MOVQ    RUNBLOCK, RUN              ; 1835
                                            65  D5 00008 1$:     TSTL    (P)                        ; 1837
                                            03  14 0000A         BGTR    2$
                                         009F   31 0000C         BRW     9$
                                     52  10     A4  D0 0000F 2$:  MOVL    16(RUN), RDB               ; 1846
                                     56  04     A2  D0 00013      MOVL    4(RDB), WFB                ; 1847
                                 5A      65  F7  8F  78 00017      ASHL    #-9, (P), R10              ; 1849
                                 50  08  A2  0C  A2  C1 0001C      ADDL3   12(RDB), 8(RDB), R0        ; 1850
                                     50  08  A4  C2 00022         SUBL2   8(RUN), R0
                                     50     5A  D1 00026         CMPL    R10, R0
                                            03  1B 00029         BLEQU   3$
                                     5A     50  D0 0002B         MOVL    R0, R10
                                 0000007F   8F  5A  D1 0002E 3$:  CMPL    R10, #127                  ; 1848
                                            04  1B 00035         BLEQU   4$
                                     5A  7F  8F  9A 00037         MOVZBL  #127, R10
                                     53     5A  D0 0003B 4$:     MOVL    R10, SIZE
                                            1C  12 0003E         BNEQ    6$                         ; 1853
```

```
                                    10  DD 00040          PUSHL   #16                                        ; 1856
                                24  A4  9F 00042          PUSHAB  36(RUN)                                    ;
                            FCE8    CF  02 FB 00045       CALLS   #2, FREE_RDB                               ;
                              10    A4  24 A4 DO 0004A    MOVL    36(RUN), 16(RUN)                           ; 1857
                                52  10  A4 DO 0004F      MOVL    16(RUN), RDB                               ; 1858
                                    59  13 00053          BEQL    9$                                         ; 1859
                              08    A4  08 A2 DO 00055    MOVL    8(RDB), 8(RUN)                             ; 1860
                                    AC  11 0005A 5$:      BRB     1$                                         ; 1852
                                    7E  7C 0005C 6$:      CLRQ    -(SP)                                      ; 1877
                                    7E  D4 0005E          CLRL    -(SP)                                      ;
                                08  A4  DD 00060          PUSHL   8(RUN)                                     ;
                        52          53  09 78 00063      ASHL    #9, SIZE, R2                               ;
                                    52  DD 00067          PUSHL   R2                                         ;
                                04  A5  DD 00069          PUSHL   4(P)                                       ;
                                    7E  7C 0006C          CLRQ    -(SP)                                      ;
                                20  AE  9F 0006E          PUSHAB  IOSB                                       ;
                                    31  DD 00071          PUSHL   #49                                        ;
                                7E  1E  A6 3C 00073       MOVZWL  30(WFB), -(SP)                             ;
                                00E0    CB DD 00077       PUSHL   224(CTX)                                   ;
                    00000000G   00      0C FB 0007B       CALLS   #12, SYS$QIOW                              ;
                                06      50 E9 00082       BLBC    STATUS, 7$                                 ; 1878
                                50      6E 3C 00085       MOVZWL  IOSB, STATUS                               ;
                                16      50 E8 00088       BLBS    STATUS, 8$                                 ; 1880
                                        50 DD 0008B 7$:   PUSHL   STATUS                                     ; 1883
                                04      A6 9F 0008D       PUSHAB  4(WFB)                                     ;
                                        01 DD 00090       PUSHL   #1                                         ;
                                001C10B2    8F DD 00092   PUSHL   #1839282                                   ;
                    00000000G   00          04 FB 00098   CALLS   #4, SOR$$ERROR                             ;
                                            10 11 0009F   BRB     10$                                        ;
                              08    A4          53 C0 000A1 8$:   ADDL2   SIZE, 8(RUN)                       ; 1892
                                    65          52 C2 000A5       SUBL2   R2, (P)                            ; 1897
                              04    A5          52 C0 000A8       ADDL2   R2, 4(P)                           ; 1898
                                                AC 11 000AC       BRB     5$                                 ; 1837
                                    50          01 D0 000AE 9$:   MOVL    #1, R0                             ; 1910
                                    5E          08 C0 000B1 10$:  ADDL2   #8, SP                             ; 1911
                                    5A          8E D0 000B4       MOVL    (SP)+, R10                         ;
                                                   05 000B7       RSB                                        ;

; Routine Size:  184 bytes,    Routine Base:  SOR$RO_CODE + 0775
```

```
;  1866     1912  1 GLOBAL ROUTINE SOR$$WORK_READ                  ! Read from a work file
;  1867     1913  1         (
;  1868     1914  1         RUNBLOCK:       REF RUN_BLOCK    ! Address of run description block
;  1869     1915  1         ):      JSB_INPUT =
;  1870     1916  1 !++
;  1871     1917  1 !
;  1872     1918  1 ! FUNCTIONAL DESCRIPTION:
;  1873     1919  1 !
;  1874     1920  1 !       This routine reads from a work file.
;  1875     1921  1 !
;  1876     1922  1 ! FORMAL PARAMETERS:
;  1877     1923  1 !
;  1878     1924  1 !       RUNBLOCK        Address of run description block
;  1879     1925  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1880     1926  1 !
;  1881     1927  1 ! IMPLICIT INPUTS:
;  1882     1928  1 !
;  1883     1929  1 !       NONE
;  1884     1930  1 !
;  1885     1931  1 ! IMPLICIT OUTPUTS:
;  1886     1932  1 !
;  1887     1933  1 !       NONE
;  1888     1934  1 !
;  1889     1935  1 ! ROUTINE VALUE:
;  1890     1936  1 !
;  1891     1937  1 !       Address of record, or zero if end-of-run.
;  1892     1938  1 !
;  1893     1939  1 ! SIDE EFFECTS:
;  1894     1940  1 !
;  1895     1941  1 !       NONE
;  1896     1942  1 !--
;  1897     1943  2     BEGIN
;  1898     1944  2     EXTERNAL REGISTER
;  1899     1945  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  1900     1946  2     LOCAL
;  1901     1947  2         RUN:    REF RUN_BLOCK,
;  1902     1948  2         LEN;                                    ! Length of record
;  1903     1949  2     LOCAL
;  1904     1950  2         IOSB:   VECTOR[4,WORD];
;  1905     1951  2
;  1906     1952  2     RUN = RUNBLOCK[BASE_];
;  1907     1953  2
;  1908     1954  2
;  1909     1955  2     ! If we were invoked for a merge, call MRG_WORK_READ to get the record
;  1910     1956  2     !
;  1911     1957  2     IF .CTX[COM_MERGE]
;  1912     1958  2     THEN
;  1913     1959  2         RETURN MRG_WORK_READ(RUN[BASE_]);
;  1914     1960  2
;  1915     1961  2
;  1916     1962  2     RUN[RUN_COUNT] = .RUN[RUN_COUNT] - 1;       ! One less record
;  1917     1963  2     IF .RUN[RUN_COUNT] LSS 0
;  1918     1964  2     THEN
;  1919     1965  2         RETURN DELRUN(RUN[BASE_]);              ! Return now if no more records
;  1920     1966  2
;  1921     1967  2
;  1922     1968  2     ! Set the length of the record to be read.
```

```
;  1923          1969  2     !
;  1924          1970  2     LEN = .CTX[COM_LRL_INT];
;  1925          1971  2
;  1926          1972  2
;  1927          1973  2     ! If everything fine, return now with the address of the record.
;  1928          1974  2     !
;  1929          1975  2     IF (RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] - .LEN) GEQ 0
;  1930          1976  2     THEN
;  1931          1977  3         BEGIN
;  1932          1978  3         RUN[RUN_BUFF1] = .RUN[RUN_BUFF1] + .LEN;        ! Next free byte
;  1933          1979  3         RETURN .RUN[RUN_BUFF1] - .LEN;
;  1934          1980  3         END
;  1935          1981  2     ELSE
;  1936          1982  2         RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] + .LEN;        ! Reset
;  1937          1983  2
;  1938          1984  2
;  1939          1985  2     ! Not all of this record is in the buffer.
;  1940          1986  2     !
;  1941          1987  2     ! We move the first part of the record backwards in the buffer, so that
;  1942          1988  2     ! the first part ends at a page boundary.  Then we request that the rest
;  1943          1989  2     ! of the buffer be filled.
;  1944          1990  2     !
;  1945          1991  3     BEGIN
;  1946          1992  3     LOCAL
;  1947          1993  3         TMP,
;  1948          1994  3         P:      VECTOR[2];
;  1949          1995  3
;  1950          1996  3     ! Compute the address of the next free byte following the copied string.
;  1951          1997  3     ! Then actually copy the first part of the string.
;  1952          1998  3     !
;  1953          1999  3     P[1] = .RUN[RUN_BUFFER] + ROUND_(.RUN[RUN_BUFF0], COM_K_BPERPAGE);
;  1954          2000  3     TMP = .P[1] - .RUN[RUN_BUFF0];
;  1955          2001  3     CH$MOVE(.RUN[RUN_BUFF0], .RUN[RUN_BUFF1], .TMP);
;  1956          2002  3
;  1957          2003  3     ! Set BUFF0/1 to the length/address of the start of the (moved) record.
;  1958          2004  3     !
;  1959          2005  3     RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] + .RUN[RUN_BUFF1] - .TMP;
;  1960          2006  3     RUN[RUN_BUFF1] = .TMP;
;  1961          2007  3
;  1962          2008  3     ! Set up for the read
;  1963          2009  3     !
;  1964          2010  3     P[0] = .RUN[RUN_BUFF1] + .RUN[RUN_BUFF0] - .P[1];
;  1965          2011  3
;  1966          2012  3     ! Fill the buffer
;  1967          2013  3     !
;  1968          2014  3     IF NOT FILL(RUN[BASE_], P[0])
;  1969          2015  3     THEN
;  1970          2016  3         RETURN DELRUN(RUN[BASE_]);      ! Return 0 to indicate end of file
;  1971          2017  3
;  1972          2018  3     RUN[RUN_BUFF0] = .RUN[RUN_BUFF0] - .LEN;
;  1973          2019  3     RUN[RUN_BUFF1] = .RUN[RUN_BUFF1] + .LEN;
;  1974          2020  3     RETURN .RUN[RUN_BUFF1] - .LEN;
;  1975          2021  2     END;
;  1976          2022  2
;  1977          2023  1     END;
```

```
                                5A  DD 00000 SOR$$WORK_READ::
                                               PUSHL   R10                             ; 1912
                             5E    1C  C2 00002          SUBL2   #28, SP
                                   59  DD 00005          PUSHL   RUNBLOCK             ; 1952
                          5C       AB  95 00007          TSTB    92(CTX)              ; 1957
                                   08  18 0000A          BGEQ    1$
                             59    6E  D0 0000C          MOVL    RUN, R9              ; 1959
                                FDA5   30 0000F          BSBW    MRG_WORK_READ
                                   3A  11 00012          BRB     3$
                       50        6E    20  C1 00014 1$:  ADDL3   #32, RUN, R0        ; 1962
                                   60  D7 00018          DECL    (R0)
                       50        6E    20  C1 0001A      ADDL3   #32, RUN, R0        ; 1963
                                   60  D5 0001E          TSTL    (R0)
                                   03  18 00020          BGEQ    2$
                                 0086  31 00022          BRW     5$
                    0C   AE      0088    CB  3C 00025 2$:  MOVZWL  136(CTX), LEN      ; 1970
                       50        6E    18  C1 0002B      ADDL3   #24, RUN, R0        ; 1975
                    04   AE            60  9E 0002F      MOVAB   (R0), 4(SP)
                    04   BE      0C    AE  C2 00033      SUBL2   LEN, @4(SP)
                                   16  19 00038          BLSS    4$
                       50        6E    1C  C1 0003A      ADDL3   #28, RUN, R0        ;978
                                 60      0C  AE C0 0003E  ADDL2   LEN, (R0)
                       50        6E    1C  C1 00042      ADDL3   #28, RUN, R0        ; 1979
                       59        60    0C  AE C3 00046   SUBL3   LEN, (R0), R9
                                 50        59  D0 0004B    MOVL    R9, R0
                                   74  11 0004E 3$:    BRB     7$
                    04   BE      0C    AE  C0 00050 4$:  ADDL2   LEN, @4(SP)         ; 1982
                 50      04   BE  000001FF    8F  C1 00055  ADDL3   #511, @4(SP), R0  ; 1999
                                 50      01FF    8F  AA 0005E  BICW2   #511, R0
                 51              6E            14  C1 00063  ADDL3   #20, RUN, R1
             14  AE              50            61  C1 00067  ADDL3   (R1), R0, P+4
                 56      14   AE      04    BE  C3 0006C      SUBL3   @4(SP), P+4, TMP  ; 2000
                 50           6E            1C  C1 00072      ADDL3   #28, RUN, R0     ; 2001
                          08  AE            60  9E 00076      MOVAB   (R0), 8(SP)
                              59      08    BE  D0 0007A      MOVL    @8(SP), R9
                 66           69      04    BE  28 0007E      MOVC3   @4(SP), (R9), (TMP)
                 50       04  BE      08    BE  C1 00083      ADDL3   @8(SP), @4(SP), R0  ; 2005
             04  BE           50            56  C3 00089      SUBL3   TMP, R0, @4(SP)
                          08  BE            56  D0 0008E      MOVL    TMP, @8(SP)      ; 2006
                 50       08  BE      04    BE  C1 00092      ADDL3   @4(SP), @8(SP), R0  ; 2010
             10  AE           50      14    AE  C3 00098      SUBL3   P+4, R0, P
                              5A      10    AE  9E 0009E      MOVAB   P, R10           ; 2014
                              59            6E  D0 000A2      MOVL    RUN, R9
                                         FEA0   30 000A5      BSBW    FILL
                                 09            50  E8 000A8  BLBS    R0, 6$
                                               6E  DD 000AB 5$:  PUSHL   RUN          ; 2016
                          0000V  CF            01  FB 000AD  CALLS   #1, DELRUN
                                               10  11 000B2  BRB     7$
                       04   BE      0C    AE  C2 000B4 6$:  SUBL2   LEN, @4(SP)      ; 2018
                       08   BE      0C    AE  C0 000B9      ADDL2   LEN, @8(SP)      ; 2019
                 50    08   BE      0C    AE  C3 000BE      SUBL3   LEN, @8(SP), R0  ; 2020
                              5E            20  C0 000C4 7$:  ADDL2   #32, SP        ; 2023
                              5A            8E  D0 000C7      MOVL    (SP)+, R10
                                            05  00CA         RSB
```

; Routine Size: 203 bytes,   Routine Base: SOR$RO_CODE + 082D

```
;  1979     2024  1 GLOBAL ROUTINE POSITION                          ! Position to start of run
;  1980     2025  1         (
;  1981     2026  1         RUNBLOCK:       REF RUN_BLOCK           ! Run to be positioned to
;  1982     2027  1         ):      CAL_CTXREG NOVALUE =
;  1983     2028  1 !++
;  1984     2029  1 !
;  1985     2030  1 ! FUNCTIONAL DESCRIPTION:
;  1986     2031  1 !
;  1987     2032  1 !       This routine positions to a run in a work file.
;  1988     2033  1 !       It also starts reading that run.
;  1989     2034  1 !
;  1990     2035  1 ! FORMAL PARAMETERS:
;  1991     2036  1 !
;  1992     2037  1 !       RUNBLOCK        Address of run description block to be positioned to
;  1993     2038  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  1994     2039  1 !
;  1995     2040  1 ! IMPLICIT INPUTS:
;  1996     2041  1 !
;  1997     2042  1 !       NONE
;  1998     2043  1 !
;  1999     2044  1 ! IMPLICIT OUTPUTS:
;  2000     2045  1 !
;  2001     2046  1 !       NONE
;  2002     2047  1 !
;  2003     2048  1 ! ROUTINE VALUE:
;  2004     2049  1 !
;  2005     2050  1 !       NONE
;  2006     2051  1 !
;  2007     2052  1 ! SIDE EFFECTS:
;  2008     2053  1 !
;  2009     2054  1 !       NONE
;  2010     2055  1 !--
;  2011     2056  2     BEGIN
;  2012     2057  2     EXTERNAL REGISTER
;  2013     2058  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  2014     2059  2     LOCAL
;  2015     2060  2         RUN:    REF RUN_BLOCK,                  ! Pointer to run block
;  2016     2061  2         STATUS;
;  2017     2062  2     LOCAL
;  2018     2063  2         P:      VECTOR[2],
;  2019     2064  2         IOSB:   VECTOR[4,WORD];
;  2020     2065  2
;  2021     2066  2
;  2022     2067  2     ! Get a pointer to the run description block, and a buffer for the run
;  2023     2068  2     !
;  2024     2069  2     RUN = RUNBLOCK[BASE_];
;  2025     2070  2     GET_BUFFER(RUN[BASE_]);
;  2026     2071  2
;  2027     2072  2
;  2028     2073  2     ! Set up for the read.
;  2029     2074  2     ! Set the starting virtual block address of this run.
;  2030     2075  2     ! Set the length and address of the buffer.
;  2031     2076  2     !
;  2032     2077  2     RUN[RUN_RDB_CURR] = .RUN[RUN_RDB];
;  2033     2078  2     RUN[RUN_VBN] = .BBLOCK[.RUN[RUN_RDB_CURR],RDB_VBN];
;  2034     2079  2
;  2035     2080  2     P[0] = MINU(.RUN[RUN_BUFF0], .RUN[RUN_SIZE]^LN2_(COM_K_BPERPAGE));
```

```
;  2036          2081  2     P[1] = .RUN[RUN_BUFFER];
;  2037          2082  2
;  2038          2083  2     ! Fill the buffer
;  2039          2084  2     !
;  2040          2085  2     IF NOT FILL(RUN[BASE_], P[0])
;  2041          2086  2     THEN
;  2042          2087  3         BEGIN
;  2043          2088  3         RUN[RUN_COUNT] = 0;              ! Indicate end of file
;  2044          2089  3         RETURN;
;  2045          2090  2         END;
;  2046          2091  2
;  2047          2092  1     END;


                                    06FC 00000        .ENTRY  POSITION, Save R2,R3,R4,R5,R6,R7,R9,R10  ; 2024
                              5E        10  C2 00002  SUBL2   #16, SP                                  ;
                              57    04  AC  D0 00005  MOVL    RUNBLOCK, RUN                            ; 2069
                                        57  DD 00009  PUSHL   RUN                                      ; 2070
                          F6F8  CF        01  FB 0000B  CALLS   #1, GET_BUFFER                        ;
                            10  A7    24  A7  D0 00010  MOVL    36(RUN), 16(RUN)                       ; 2077
                              50    10  A7  D0 00015  MOVL    16(RUN), R0                              ; 2078
                            08  A7    08  A0  D0 00019  MOVL    8(R0), 8(RUN)                          ;
                50          0C  A7        09  78 0001E  ASHL    #9, 12(RUN), R0                          ; 2080
                              59    18  A7  D0 00023  MOVL    24(RUN), R9                              ;
                              50        59  D1 00027  CMPL    R9, R0                                   ;
                                        03  1B 0002A  BLEQU   1$                                       ;
                              59        50  D0 0002C  MOVL    R0, R9                                   ;
                        08    AE        59  D0 0002F  1$:     MOVL    R9, P                            ;
                        0C    AE  14    A7  D0 00033  MOVL    20(RUN), P+4                             ; 2081
                              5A  08    AE  9E 00038  MOVAB   P, R10                                   ; 2085
                              59        57  D0 0003C  MOVL    RUN, R9                                  ;
                                    FE3B  30 0003F  BSBW    FILL                                     ;
                              03        50  E8 00042  BLBS    R0, 2$                                   ;
                                    20  A7  D4 00045  CLRL    32(RUN)                                  ; 2088
                                        04 00048  2$:     RET                                          ; 2092

; Routine Size:  73 bytes,    Routine Base:  SOR$RO_CODE + 08F8
```

```
;  2049          2093  1 ROUTINE DELRUN                                . Indicate this run is through
;  2050          2094  1         (
;  2051          2095  1         RUNBLOCK:       REF RUN_BLOCK           ! Run description block
;  2052          2096  1         ):      CAL_CTXREG =
;  2053          2097  1 !++
;  2054          2098  1 !
;  2055          2099  1 ! FUNCTIONAL DESCRIPTION:
;  2056          2100  1 !
;  2057          2101  1 !       This routine indicates that processing for a run is complete.
;  2058          2102  1 !
;  2059          2103  1 ! FORMAL PARAMETERS:
;  2060          2104  1 !
;  2061          2105  1 !       RUNBLOCK        Address of run description block of run to delete
;  2062          2106  1 !       CTX             Longword pointing to work area (passed in COM_REG_CTX)
;  2063          2107  1 !
;  2064          2108  1 ! IMPLICIT INPUTS:
;  2065          2109  1 !
;  2066          2110  1 !       NONE
;  2067          2111  1 !
;  2068          2112  1 ! IMPLICIT OUTPUTS:
;  2069          2113  1 !
;  2070          2114  1 !       NONE
;  2071          2115  1 !
;  2072          2116  1 ! ROUTINE VALUE:
;  2073          2117  1 !
;  2074          2118  1 !       Zero.
;  2075          2119  1 !
;  2076          2120  1 ! SIDE EFFECTS:
;  2077          2121  1 !
;  2078          2122  1 !       NONE
;  2079          2123  1 !--
;  2080          2124  2     BEGIN
;  2081          2125  2     EXTERNAL REGISTER
;  2082          2126  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  2083          2127  2     LOCAL
;  2084          2128  2         P:      REF RUN_BLOCK,
;  2085          2129  2         Q:      REF VECTOR[1],
;  2086          2130  2         R:      REF RDB_BLOCK,
;  2087          2131  2         STATUS;
;  2088          2132  2
;  2089          2133  2
;  2090          2134  2     ! One less outstanding run
;  2091          2135  2     !
;  2092          2136  2     CTX[COM_RUNS] = .CTX[COM_RUNS] - 1;         ! One less run
;  2093          2137  2
;  2094          2138  2     ! Free all RDBs connected to this RUN block
;  2095          2139  2     !
;  2096          2140  2     Q = RUNBLOCK[RUN_RDB];
;  2097          2141  2     RUNBLOCK[RUN_RDB_CURR] = 0;
;  2098          2142  2     MOVE_ALL_RDBS_( Q[0], WFB_FREE );
;  2099          2143  2
;  2100          2144  2     ! Find the run description block in the list of active runs,
;  2101          2145  2     ! and remove it from this list.
;  2102          2146  2     !
;  2103          2147  2     Q = CTX[S_RUN_INFO];
;  2104          2148  2     WHILE (P = .Q[0]) NEQ 0 DO
;  2105          2149  3         BEGIN
```

```
;  2106          2150  3          IF P[BASE_] EQL RUNBLOCK[BASE_] THEN EXITLOOP;
;  2107          2151  3          Q = P[RUN_NEXT];
;  2108          2152  2          END;
;  2109          2153  2      IF P[BASE_] EQL 0
;  2110          2154  2      THEN
;  2111          2155  2          RETURN SOR$$FATAL(SOR$_SHR_BADLOGIC);   ! We couldn't find it
;  2112          2156  2
;  2113          2157  2      ! Free the buffer associated with this run.
;  2114          2158  2      ! Unlink and deallocate the run block.
;  2115          2159  2      !
;  2116          2160  2      FREE_BUFFER(P[BASE_]);                          ! Free this run's buffer
;  2117          2161  2      REMLR_(Q[0], P, RUN_NEXT, RUN_K_SIZE);          ! Unlink this RUN block
;  2118          2162  2
;  2119          2163  2      ! Return zero, since most of our callers want to immediately return 0
;  2120          2164  2      ! after calling us.
;  2121          2165  2      !
;  2122          2166  2      RETURN 0;
;  2123          2167  1      END;


                                        001C 00000 DELRUN:  .WORD   Save R2,R3,R4                   ; 2093
                                7A  AB  B7 00002            DECW    122(CTX)                        ; 2136
                            53  04  AC  D0 00005            MOVL    RUNBLOCK, R3                    ; 2140
                            52  24  A3  9E 00009            MOVAB   36(R3), Q
                                10  A3  D4 0000D            CLRL    16(R3)                          ; 2141
                                    62  D5 00010 1$:        TSTL    (Q)                             ; 2142
                                    0B  13 00012            BEQL    2$
                                    10  DD 00014            PUSHL   #16
                                    52  DD 00016            PUSHL   Q
                                FB49    CF  02  FB 00018    CALLS   #2, FREE_RDB
                                    F1  11 0001D            BRB     1$
                                    52  00D8  CB  9E 0001F 2$:  MOVAB   216(R11), Q                ; 2147
                                    54  62  D0 00024 3$:      MOVL    (Q), P                      ; 2148
                                        0A  13 00027        BEQL    4$
                                    53  54  D1 00029        CMPL    P, R3                       ; 2150
                                        05  13 0002C        BEQL    4$
                                    52  54  D0 0002E        MOVL    P, Q                        ; 2151
                                        F1  11 00031        BRB     3$                          ; 2148
                                        54  D5 00033 4$:    TSTL    P                           ; 2153
                                        0E  12 00035        BNEQ    5$
                                001C1124    8F  DD 00037    PUSHL   #1839396                    ; 2155
                    00000000G   00          01  FB 0003D   CALLS   #1, SOR$$ERROR
                                            04  00044     RET
                                        54  DD 00045 5$:    PUSHL   P                           ; 2160
                                F701    CF  01  FB 00047    CALLS   #1, FREE_BUFFER
                                        64  DD 0004C        PUSHL   (P)                         ; 2161
                                        52  DD 0004E        PUSHL   Q
                                        28  DD 00050        PUSHL   #40
                    00000000G   00          03  FB 00052    CALLS   #3, SOR$$DEALLOCATE
                                        50  D4 00059        CLRL    R0                          ; 2166
                                            04  0005B      RET                                 ; 2167

; Routine Size:  92 bytes,    Routine Base:  SOR$RO_CODE + 0941
```

```
;  2125          2168  1 ROUTINE WORK_OPEN
;  2126          2169  1         (
;  2127          2170  1         FNS:    REF VECTOR[2]           ! Work file name string
;  2128          2171  1         ) : CAL_CTXREG NOVALUE =
;  2129          2172  1
;  2130          2173  1 !++
;  2131          2174  1 !
;  2132          2175  1 ! FUNCTIONAL DESCRIPTION:
;  2133          2176  1 !
;  2134          2177  1 !       This routine creates/opens the work files.
;  2135          2178  1 !
;  2136          2179  1 ! FORMAL PARAMETERS:
;  2137          2180  1 !
;  2138          2181  1 !       FNS     File name string of work file to be opened (optional)
;  2139          2182  1 !               If not present, all COM_WORK_FILES work files are opened.
;  2140          2183  1 !               If present, only one work file is opened.
;  2141          2184  1 !       CTX     Longword pointing to work area (passed in COM_REG_CTX)
;  2142          2185  1 !
;  2143          2186  1 ! IMPLICIT INPUTS:
;  2144          2187  1 !
;  2145          2188  1 !       NONE
;  2146          2189  1 !
;  2147          2190  1 ! IMPLICIT OUTPUTS:
;  2148          2191  1 !
;  2149          2192  1 !       NONE
;  2150          2193  1 !
;  2151          2194  1 ! ROUTINE VALUE:
;  2152          2195  1 !
;  2153          2196  1 !       NONE
;  2154          2197  1 !
;  2155          2198  1 ! SIDE EFFECTS:
;  2156          2199  1 !
;  2157          2200  1 !       CTX[S_BUF_SIZE] is calculated.
;  2158          2201  1 !--
;  2159          2202  2     BEGIN
;  2160          2203  2     EXTERNAL REGISTER
;  2161          2204  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  2162          2205  2     BUILTIN
;  2163          2206  2         NULLPARAMETER;
;  2164          2207  2     LOCAL
;  2165          2208  2         FAB:    $FAB_DECL,                          ! FAB for the work file
;  2166          2209  2         NAM:    $NAM_DECL VOLATILE,                 ! NAM for the work file
;  2167          2210  2         FNA:    VECTOR[NAM$C_MAXRSSLCL,BYTE],       ! File name string
;  2168          2211  2         WFB:    REF WFB_BLOCK,                      ! Pointer to WFB for work files
;  2169          2212  2         WFB_PTR:        REF VECTOR[1,LONG],         ! Addr of pointer to WFB
;  2170          2213  2         XABPRO: $XABPRO_DECL,                       ! XAB for file protection
;  2171          2214  2         STATUS;                                     ! Status
;  2172          2215  2
;  2173          2216  2
;  2174          2217  2     ! Set the default extend amount to half the tree size
;  2175          2218  2     !
;  2176          2219  2     CTX[S_WRK_DEQ] = MAXU(
;  2177          2220  2         .CTX[COM_TREE_LEN] ^ -LN2 (COM_K_BPERBLOCK*2),
;  2178          2221  2         TUN_K_BUFSIZE / COM_K_BPERBLOCK);
;  2179          2222  2
;  2180          2223  2     ! Get an event flag to use for work file I/O
;  2181          2224  2     !
```

```
;  2182          2225  2      %IF NOT HOSTILE %THEN
;  2183          2226  2      STATUS = LIB$GET_EF(CTX[S_WRK_EFN]);
;  2184          2227  2      IF NOT .STATUS
;  2185          2228  2      THEN
;  2186          2229  2          RETURN SOR$$ERROR(SOR$_SHR_SYSERROR, 0, .STATUS);
;  2187          2230  2      %FI
;  2188          2231  2
;  2189        P 2232  2      $FAB_INIT(
;  2190        P 2233  2          FAB = FAB[BASE_],
;  2191        P 2234  2          NAM = NAM[BASE_],
;  2192        P 2235  2          FNA = FNA[0],
;  2193        P 2236  2  !       FNS = 0,                    ! Set below
;  2194        P 2237  2          DNA = UPLIT BYTE(STR_DEF_WORKFILE),
;  2195        P 2238  2          DNS = %CHARCOUNT(STR_DEF_WORKFILE),
;  2196        L 2239  2  %IF TUN_K_WRK_PREALL
;  2197        U 2240  2  %THEN
;  2198        U 2241  2          ALQ = 2 * .CTX[S_WRK_DEQ]                   ! Tree size
;  2199        U 2242  2                / .CTX[COM_WRK_FILES],                ! Divided by number of files
;  2200        P 2243  2  %FI
;  2201        P 2244  2          DFQ = .CTX[S_WRK_DEQ],                      ! Default extension quantity
;  2202        P 2245  2  !       BKS = 0,                    ! Only for relative or indexed files
;  2203        P 2246  2  !       BLS = 0,                    ! Only for magnetic tape files
;  2204        P 2247  2          FAC = <BIO,DEL,GET,PUT,UPD>,
;  2205        P 2248  2          FOP = <CBT,DFW,DLT,SUP,TMD,TMP,SQO,UFO>,        ! TMD,TMP???
;  2206        P 2249  2          FSZ = 0,
;  2207        P 2250  2          MRN = 0,
;  2208        P 2251  2          MRS = COM_K_BPERBLOCK,
;  2209        P 2252  2          ORG = SEQ,
;  2210        P 2253  2  !       RAT = 0,                    ! No special record attributes
;  2211        P 2254  2          RFM = FIX,                  ! Fixed format records
;  2212        P 2255  2          RTV = 10,                   ! 10 retrieval pointers
;  2213        P 2256  2  !       SHR = GET,                  ! Allow others to read this file
;  2214          2257  2          XAB = XABPRO[BASE_]);       ! File protection XAB
;  2215        P 2258  2      $NAM_INIT(
;  2216        P 2259  2          NAM = NAM[BASE_],
;  2217        P 2260  2          ESS = NAM$C_MAXRSSLCL,
;  2218        P 2261  2          ESA = FNA[0],
;  2219        P 2262  2          RSS = NAM$C_MAXRSSLCL,
;  2220          2263  2          RSA = FNA[0]);
;  2221          2264  2
;  2222          2265  2
;  2223          2266  2      WFB_PTR = CTX[S_WRK_WFB];
;  2224          2267  2      WHILE .WFB_PTR[0] NEQ 0 DO WFB_PTR = BBLOCK[.WFB_PTR[0],WFB_NEXT];
;  2225          2268  2      DECR I FROM .CTX[COM_WRK_FILES]-1 TO 0 DO
;  2226          2269  3          BEGIN
;  2227          2270  3          STRUCTURE
;  2228          2271  3              BVECTOR[I;N,EXT=0] = [N](BVECTOR+I)<0,8,EXT>;
;  2229          2272  3          !
;  2230          2273  3          ! Allocate WFB and link into list (advance to next file at end of loop)
;  2231          2274  3          ! These are put at the end of the list so that SORTWORK0 is used first,
;  2232          2275  3          ! followed by SORTWORK1, et cetera.
;  2233          2276  3          !
;  2234          2277  3          WFB = SOR$$ALLOCATE(WFB_K_SIZE);
;  2235          2278  3          WFB_PTR[0] = WFB[WFB_NEXT];     ! Store address of the next pointer
;  2236          2279  3
;  2237          2280  3          ! Get the name of this file
;  2238          2281  3          !
```

```
;  2239          2282  3          IF NOT NULLPARAMETER(1)
;  2240          2283  3          THEN
;  2241          2284  4              BEGIN
;  2242          2285  4              CH$MOVE(
;  2243          2286  4                  (FAB[FAB$B_FNS] = MINU(.FNS[0], %ALLOCATION(FNA))),
;  2244          2287  4                  .FNS[1], FNA[0]);
;  2245          2288  4              END
;  2246          2289  3          ELIF .I GEQU .BVECTOR[CTX[COM_WF_NAMES], 0]
;  2247          2290  3          THEN
;  2248          2291  4              BEGIN
;  2249          2292  4              FAB[FAB$B_FNS] = %CHARCOUNT(STR_LOG_WORKFILE)+1;
;  2250          2293  4              CH$MOVE(
;  2251          2294  4                  %CHARCOUNT(STR_LOG_WORKFILE),
;  2252          2295  4                  UPLIT BYTE(STR_LOG_WORKFILE),
;  2253          2296  4                  FNA[0]);
;  2254          2297  4              ASSERT (%CHARCOUNT(STR_LOG_WORKNUM) GEQ MAX_WORK_FILES)
;  2255          2298  4              FNA[%CHARCOUNT(STR_LOG_WORKFILE)] = .BVECTOR[UPLIT BYTE(
;  2256          2299  4                  %EXACTSTRING(MAX_WORK_FILES,0,STR_LOG_WORKNUM)), .I];
;  2257          2300  4              END
;  2258          2301  3          ELSE
;  2259          2302  3              %IF NOT HOSTILE %THEN
;  2260          2303  4              BEGIN
;  2261          2304  4              LIBRARY 'SRC$:SRTSPC';
;  2262          2305  4              LOCAL
;  2263          2306  4                  CFT: REF CFT_TAB[1];
;  2264          2307  4              CFT = CFT_TAB[.CTX[COM_CFT_ADR],
;  2265          2308  4                  .BVECTOR[CTX[COM_WF_NAMES], .I+1], BASE_];
;  2266          2309  4              CH$MOVE(
;  2267          2310  4                  (FAB[FAB$B_FNS] = MINU(.CFT[0,CFT_CON_LEN], %ALLOCATION(FNA))),
;  2268          2311  4                  .CFT[0,CFT_CON_ADR]+.CTX[COM_CFT_ADR], FNA[0]);
;  2269          2312  3              END;
;  2270        U 2313  3              %ELSE
;  2271        U 2314  3              RETURN SOR$$FATAL(SOR$_SHR_BADLOGIC);
;  2272          2315  3              %FI
;  2273          2316  3
;  2274          2317  3          ! Clear these fields, since we are reusing the NAM, FAB and XAB blocks
;  2275          2318  3          !
;  2276          2319  3          NAM[NAM$B_RSL] = 0;
;  2277          2320  3          NAM[NAM$B_ESL] = 0;
;  2278          2321  3          FAB[FAB$W_IFI] = 0;
;  2279        P 2322  3          $XABPRO_INIT(
;  2280        P 2323  3              XAB = XABPRO[BASE_],
;  2281          2324  3              PRO = <D,RWED>);
;  2282          2325  3
;  2283          2326  3          ! Actually create the work file
;  2284          2327  3          !
;  2285        P 2328  3          STATUS = $CREATE(FAB = FAB[BASE_]
;  2286          2329  3              %IF HOSTILE_ELAN %THEN , ELAN = .CTX[COM_CTXADR] %FI );
;  2287          2330  3
;  2288          2331  3          ! Get the best file name string available into NAM$B_RSL/NAM$L_RSA
;  2289          2332  3          !
;  2290          2333  3          SOR$$BEST_FILE_NAME(FAB[BASE_], WFB[WFB_NAME]);
;  2291          2334  3
;  2292          2335  3          ! Check status
;  2293          2336  3          !
;  2294          2337  3          IF NOT .FAB[FAB$L_STS]
;  2295          2338  3          THEN
```

```
;  2296          2339  4              BEGIN
;  2297          2340  4              SOR$$ERROR(SOR$_SHR_OPENOUT AND NOT STS$M_SEVERITY OR STS$K_ERROR,
;  2298          2341  4                  1, WFB[WFB_NAME], .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
;  2299          2342  4              END
;  2300          2343  3          ELIF NOT .BBLOCK[FAB[FAB$L_DEV], DEV$V_RND] OR .NAM[NAM$V_NODE]
;  2301          2344  3          THEN
;  2302          2345  4              BEGIN
;  2303          2346  4              !
;  2304          2347  4              ! Device must be local and random access.
;  2305          2348  4              !
;  2306        P 2349  4              $DASSGN(CHAN=.FAB[FAB$L_STV]
;  2307          2350  4                  %IF HOSTILE_ELAN %THEN , ELAN = .CTX[COM_CTXADR] %FI );
;  2308          2351  4              SOR$$ERROR(SOR$_WORK_DEV, 1, WFB[WFB_NAME]);
;  2309          2352  4              END
;  2310          2353  3          ELSE
;  2311          2354  4              BEGIN
;  2312          2355  4              !
;  2313          2356  4              ! Save the channel number (returned in STV)
;  2314          2357  4              ! Create a large hole describing available space.
;  2315          2358  4              !
;  2316          2359  4              LOCAL
;  2317          2360  4                  RDB:    REF RDB_BLOCK;
;  2318          2361  4              WFB[WFB_CHAN] = .FAB[FAB$L_STV];
;  2319          2362  4              RDB = SOR$$ALLOCATE(RDB_K_SIZE);
;  2320          2363  4              INSLH (WFB[WFB_NOUSE], RDB, RDB_NEXT);
;  2321          2364  4              RDB[RDB_WFB] = WFB[BASE_];              ! Reference the WFB
;  2322          2365  4              RDB[RDB_VBN] = 1;                       ! First virtual block number
;  2323          2366  4              RDB[RDB_SIZE] = -1;                     ! A very large hole
;  2324          2367  3              END;
;  2325          2368  3
;  2326          2369  3
;  2327          2370  3          ! Check for an error
;  2328          2371  3          !
;  2329          2372  3          IF .WFB[WFB_CHAN] EQL 0
;  2330          2373  3          THEN
;  2331          2374  4              BEGIN
;  2332          2375  4              !
;  2333          2376  4              ! Deallocate and unlink
;  2334          2377  4              !
;  2335          2378  4              SOR$$FREE_FILE_NAME(WFB[WFB_NAME]); ! Free name string
;  2336          2379  4              REMLH (WFB_PTR[0], WFB, WFB_NEXT, WFB_K_SIZE);
;  2337          2380  4              CTX[COM_WRK_FILES] = .CTX[COM_WRK_FILES] - 1;
;  2338          2381  4              END
;  2339          2382  3          ELSE
;  2340          2383  4              BEGIN
;  2341          2384  4              IF NOT NULLPARAMETER(1)
;  2342          2385  4              THEN
;  2343          2386  4                  SOR$$ERROR(SOR$_USE_ALT,
;  2344          2387  4                      2, UPLIT(%ASCIC 'work'), WFB[WFB_NAME]);
;  2345          2388  4              WFB_PTR = WFB[WFB_NEXT];                ! Advance to next WFB
;  2346          2389  3              END;
;  2347          2390  2          END;
;  2348          2391  2
;  2349          2392  2      ! If we didn't get enough work files, abort.
;  2350          2393  2      !
;  2351          2394  2      IF NOT NULLPARAMETER(1) THEN RETURN;
;  2352          2395  2      IF .CTX[COM_WRK_FILES] LSS MIN_WORK_FILES
```

```
;  2353          2396  2      THEN
;  2354          2397  3          BEGIN
;  2355          2398  3          LOCAL
;  2356          2399  3              SYS_DISK: VECTOR[2] INITIAL(
;  2357          2400  3                  %CHARCOUNT('SYS$DISK:'),
;  2358          2401  3                  UPLIT BYTE('SYS$DISK:'));
;  2359          2402  3
;  2360          2403  3          ! Try using SYS$DISK for the work files
;  2361          2404  3          !
;  2362          2405  3          CTX[COM_WRK_FILES] = MIN_WORK_FILES;
;  2363          2406  3          ASSERT (MIN_WORK_FILES EQL 1)
;  2364          2407  3          WORK_OPEN(SYS_DISK);
;  2365          2408  3          IF .CTX[COM_WRK_FILES] LSS MIN_WORK_FILES
;  2366          2409  3          THEN
;  2367          2410  3              RETURN SOR$$FATAL(SOR$_NO_WRK);
;  2368          2411  2          END;
;  2369          2412  2
;  2370          2413  2
;  2371          2414  2      ! CTX[S_BUF_SIZE] is the number of bytes to use for each buffer.
;  2372          2415  2      ! It may have been calculated earlier, but that calculation may have been
;  2373          2416  2      ! too small.  We need at least enough room for a record (rounded up to a
;  2374          2417  2      ! full number of pages) plus an extra page.
;  2375          2418  2      !
;  2376          2419  2      ASSERT_(COM_K_BPERPAGE EQL COM_K_BPERBLOCK)
;  2377          2420  2      ASSERT_(TUN_K_BUFSIZE GEQ 2*COM_K_BPERPAGE)
;  2378          2421  2      CTX[S_BUF_SIZE] = MAX(
;  2379          2422  2          TUN_K_BUFSIZE,
;  2380          2423  2          .CTX[COM_LRL_INT] + COM_K_BPERPAGE,
;  2381          2424  2          .CTX[S_BUF_SIZE]);
;  2382          2425  2      CTX[S_BUF_SIZE] = ROUND_(.CTX[S_BUF_SIZE], COM_K_BPERPAGE);
;  2383          2426  2
;  2384          2427  1      END;


52 4F 53 3A 48 43 54 41 52 43 53 24 53 59 53  0099D P.AAA:  .ASCII  \SYS$SCRATCH:SORTWORK.TMP\                ;
                  50 4D 54 2E 4B 52 4F 57 54  009AC                                                          ;
                     4B 52 4F 57 54 52 4F 53  009B5 P.AAB:  .ASCII  \SORTWORK\                                ;
                  39 38 37 36 35 34 33 32 31  009BD P.AAC:  .ASCII  \0123456789\                              ;
                                              009C7        .BLKB   1
                     00 00 00 68 72 6F 77 04  009C8 P.AAD:  .ASCII  <4>\work\<0><0><0>                       ;
                  3A 4B 53 49 44 24 53 59 53  009D0 P.AAE:  .ASCII  \SYS$DISK:\                               ;

                                              .EXTRN  SYS$CREATE, SYS$DASSGN

                                          07FC 00000 WORK_OPEN:
                                              .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                       ; 2168
                        5A 00000000G  00  9E 00002    MOVAB   SOR$$ERROR, R10                               ;
                        5E     FDF0   CE  9E 00009    MOVAB   -528(SP), SP                                  ;
                50      00C4   CB    F6   8F  78 0000E    ASHL    #-10, 196(CTX), R0                        ; 2220
                        32           50  D1 00015    CMPL    R0, #50                                       ; 2221
                                     03  1E 00018    BGEQU   1$                                            ;
                        50           32  D0 0001A    MOVL    #50, R0                                       ;
                     00DC   CB       50  D0 0001D 1$: MOVL    R0, 220(CTX)                                  ; 2219
                                 00E0   CB  9F 00022    PUSHAB  224(CTX)                                      ; 2226
                00000000G  00        01  FB 00026    CALLS   #1, LIB$GET_EF                                ;
                        59           50  D0 0002D    MOVL    R0, STATUS                                    ;
```

```
                          0E            59 E8 00030       BLBS    STATUS, 2$                          ; 2227
                                        59 DD 00033       PUSHL   STATUS                              ; 2229
                                        7E D4 00035       CLRL    -(SP)                               ;
                             001C11B4   8F DD 00037       PUSHL   #1839540                            ;
                          6A            03 FB 0003D       CALLS   #3, SOR$$ERROR                      ;
                                        04 00040          RET                                         ;
  0050  8F      00        6E            00 2C 00041 2$:   MOVC5   #0, (SP), #0, #80, $RMS_PTR         ; 2257
                                  B0    AD    00048                                                   ;
                B0   AD          5003   8F B0 0004A       MOVW    #20483, $RMS_PTR                    ;
                B4   AD      0022807C   8F D0 00050       MOVL    #2261116, $RMS_PTR+4                ;
                C4   AD          00DC   CB B0 00058       MOVW    220(CTX), $RMS_PTR+20               ;
                C6   AD                 2F 90 0005E       MOVB    #47, $RMS_PTR+22                    ;
                CC   AD                 0A B0 00062       MOVW    #10, $RMS_PTR+28                    ;
                CF   AD                 01 90 00066       MOVB    #1, $RMS_PTR+31                     ;
                D4   AD            08   AE 9E 0006A       MOVAB   XABPRO, $RMS_PTR+36                 ;
                D8   AD          FF50   CD 9E 0006F       MOVAB   NAM, $RMS_PTR+40                    ;
                DC   AD            60   AE 9E 00075       MOVAB   FNA, $RMS_PTR+44                    ;
                E0   AD          FF46   CF 9E 0007A       MOVAB   P.AAA, $RMS_PTR+48                  ;
                E5   AD                 18 90 00080       MOVB    #24, $RMS_PTR+53                    ;
                E6   AD          0200   8F B0 00084       MOVW    #512, $RMS_PTR+54                   ;
  0060  8F      00        6E            00 2C 0008A       MOVC5   #0, (SP), #0, #96, $RMS_PTR         ; 2263
                                 FF50   CD    00091                                                   ;
              FF50   CD          6002   8F B0 00094       MOVW    #24578, $RMS_PTR                    ;
              FF52   CD                 01 8E 0009B       MNEGB   #1, $RMS_PTR+2                      ;
              FF54   CD            60   AE 9E 000A0       MOVAB   FNA, $RMS_PTR+4                     ;
              FF5A   CD                 01 8E 000A6       MNEGB   #1, $RMS_PTR+10                     ;
              FF5C   CD            60   AE 9E 000AB       MOVAB   FNA, $RMS_PTR+12                    ;
                     58          00E4   CB 9E 000B1       MOVAB   228(R11), WFB_PTR                   ; 2266
                                        68 D5 000B6 3$:   TSTL    (WFB_PTR)                           ; 2267
                                        05 13 000B8       BEQL    4$                                  ;
                     58                 68 D0 000BA       MOVL    (WFB_PTR), WFB_PTR                  ;
                                        F7 11 000BD       BRB     3$                                  ;
                     57            5A   AB 9A 000BF 4$:   MOVZBL  90(CTX), I                          ; 2268
                                      014C 31 000C3       BRW     16$                                 ;
                                        20 DD 000C6 5$:   PUSHL   #32                                 ; 2277
           00000000G 00                 01 FB 000C8       CALLS   #1, SOR$$ALLOCATE                   ;
                     56                 50 D0 000CF       MOVL    R0, WFB                             ;
                     68                 56 D0 000D2       MOVL    WFB, (WFB_PTR)                      ; 2278
                                        6C 95 000D5       TSTB    (AP)                                ; 2282
                                        25 13 000D7       BEQL    7$                                  ;
                                   04   AC D5 000D9       TSTL    4(AP)                               ;
                                        20 13 000DC       BEQL    7$                                  ;
                     50            04   AC D0 000DE       MOVL    FNS, R0                             ; 2286
                     51                 60 D0 000E2       MOVL    (R0), R1                            ;
           000000FF  8F                 51 D1 000E5       CMPL    R1, #255                            ;
                                        04 1B 000EC       BLEQU   6$                                  ;
                     51            FF   8F 9A 000EE       MOVZBL  #255, R1                            ;
                E4   AD                 51 90 000F2 6$:   MOVB    R1, FAB+52                          ;
         60  AE     04   B0             51 28 000F6       MOVC3   R1, @4(R0), FNA                     ; 2287
                                        3E 11 000FC       BRB     9$                                  ; 2282
    57  0134  CB         08             00 ED 000FE 7$:   CMPZV   #0, #8, 308(CTX), I                 ; 2289
                                        14 1A 00105       BGTRU   8$                                  ;
                E4   AD                 09 90 00107       MOVB    #9, FAB+52                          ; 2292
         60  AE   FECC   CF             08 28 0010B       MOVC3   #8, P.AAB, FNA                      ; 2296
                68   AE      FECD CF47  90    00112       MOVB    P.AAC[I], FNA+8                     ; 2298
                                        21 11 00119       BRB     9$                                  ; 2288
                     50      0135 CB47  9A    0011B 8$:   MOVZBL  309(CTX)[I], R0                     ; 2308
```

```
                                50              06   C4 00121          MULL2   #6, R0                                          ;
                                50      010C    CB   C0 00124          ADDL2   268(CTX), CFT                                   ;
                                51              60   9A 00129          MOVZBL  (CFT), R1                                       ; 2310
                            E4  AD              51   90 0012C          MOVB    R1, FAB+52                                      ;
                        50  02  A0      010C    CB   C1 00130          ADDL3   268(CTX), 2(CFT), R0                            ; 2311
                    60  AE              60      51   28 00137          MOVC3   R1, (R0), FNA                                   ;
                                        FF53    CD   94 0013C 9$:      CLRB    NAM+3                                           ; 2319
                                        FF5B    CD   94 00140          CLRB    NAM+11                                          ; 2320
                                          B2    AD   B4 00144          CLRW    FAB+2                                           ; 2321
    0058    8F          00              6E      00   2C 00147          MOVC5   #0, (SP), #0, #88, $RMS_PTR                     ; 2324
                                            08  AE     0014E
                            08  AE      5813    8F   B0 00150          MOVW    #22547, $RMS_PTR                                ;
                            10  AE      FF07    8F   B0 00156          MOVW    #-249, $RMS_PTR+8                               ;
                                          B0    AD   9F 0015C          PUSHAB  FAB                                             ; 2329
                    00000000G   00              01   FB 0015F          CALLS   #1, SYS$CREATE                                  ;
                                59              50   D0 00166          MOVL    R0, STATUS                                      ;
                                52        04    A6   9E 00169          MOVAB   4(WFB), R2                                      ; 2333
                                          B8    AD   DD 0016D          PUSHL   R2                                              ;
                                          B8    AD   9F 0016F          PUSHAB  FAB                                             ;
                    00000000G   00              02   FB 00172          CALLS   #2, SOR$$BEST_FILE_NAME                         ;
                                13        B8    AD   E8 00179          BLBS    FAB+8, 10$                                      ; 2337
                                7E        B8    AD   7D 0017D          MOVQ    FAB+8, -(SP)                                    ; 2341
                                              52   DD 00181          PUSHL   R2                                              ;
                                              01   DD 00183          PUSHL   #1                                              ;
                                        001C10A2    8F   DD 00185          PUSHL   #1839266                                        ;
                                6A                  05   FB 0018B          CALLS   #5, SOR$$ERROR                                  ;
                                                    45   11 0018E          BRB     13$                                             ; 2337
                            05      F3  AD                  04   E1 00190 10$:  BBC     #4, FAB+67, 11$                         ; 2343
                            19      86  AD                  01   E1 00195      BBC     #1, NAM+54, 12$                         ;
                                          BC    AD   DD 0019A 11$:     PUSHL   FAB+12                                          ; 2350
                    00000000G   00              01   FB 0019D          CALLS   #1, SYS$DASSGN                                  ;
                                                52   DD 001A4          PUSHL   R2                                              ; 23
                                                01   DD 001A6          PUSHL   #1                                              ;
                                        001C800A    8F   DD 001A8          PUSHL   #1867786                                        ;
                                6A                  03   FB 001AE          CALLS   #3, SOR$$ERROR                                  ;
                                                    22   11 001B1          BRB     13$                                             ; 2342
                        1E  A6          BC      AD   B0 001B3 12$:     MOVW    FAB+12, 30(WFB)                                 ; 2361
                                                10   DD 001B8          PUSHL   #16                                             ; 2362
                    00000000G   00              01   FB 001BA          CALLS   #1, SOR$$ALLOCATE                               ;
                                60        14    A6   D0 001C1          MOVL    20(WFB), (RDB)                                  ; 2363
                            14  A6              50   D0 001C5          MOVL    RDB, 20(WFB)                                    ;
                            04  A0              56   D0 001C9          MOVL    WFB, 4(RDB)                                     ; 2364
                            08  A0              01   D0 001CD          MOVL    #1, 8(RDB)                                      ; 2365
                            0C  A0              01   CE 001D1          MNEGL   #1, 12(RDB)                                     ; 2366
                                          1E    A6   B5 001D5 13$:     TSTW    30(WFB)                                         ; 2372
                                                1B   12 001D8          BNEQ    14$                                             ;
                                                52   DD 001DA          PUSHL   R2                                              ; 2378
                    00000000G   00              01   FB 001DC          CALLS   #1, SOR$$FREE_FILE_NAME                         ;
                                                66   DD 001E3          PUSHL   (WFB)                                           ; 2379
                                                58   DD 001E5          PUSHL   WFB_PTR                                         ;
                                                20   DD 001E7          PUSHL   #32                                             ;
                    00000000G   00              03   FB 001E9          CALLS   #3, SOR$$DEALLOCATE                             ;
                                          5A    AB   97 001F0          DECB    90(CTX)                                         ; 2380
                                                1D   11 001F3          BRB     16$                                             ; 2372
                                                6C   95 001F5 14$:     TSTB    (AP)                                            ; 2384
                                                16   13 001F7          BEQL    15$                                             ;
                                          04    AC   D5 001F9          TSTL    4(AP)                                           ;
```

```
                                11  13 001FC          BEQL    15$                                     ;
                                52  DD 001FE          PUSHL   R2                                      ; 2387
                          FDEB  CF  9F 00200          PUSHAB  P.AAD                                   ;
                                02  DD 00204          PUSHL   #2                                      ;
                     001C81E3   8F  DD 00206          PUSHL   #1868259                                ;
                            6A      04  FB 0020C      CALLS   #4, SOR$$ERROR                          ;
                            58      56  D0 0020F 15$: MOVL    WFB, WFB_PTR                            ; 2388
                            02      57  F4 00212 16$: SOBGEQ  I, 17$                                 ; 2268
                                    03  11 00215      BRB     18$                                     ;
                                FEAC  31 00217 17$:    BRW     5$                                      ;
                                   6C  95 0021A 18$:   TSTB    (AP)                                    ; 2394
                                   05  13 0021C        BEQL    19$                                     ;
                                 04    AC  D5 0021E    TSTL    4(AP)                                   ;
                                   60  12 00221        BNEQ    23$                                     ;
                                 5A    AB  95 00223 19$: TSTB    90(CTX)                                 ; 2395
                                   23  12 00226        BNEQ    20$                                     ;
                            6E          09  D0 0C228    MOVL    #9, SYS_DISK                            ; 2397
                      04    AE    FDC8  CF  9E 0022B    MOVAB   P.AAE, SYS_DISK+4                       ;
                      5A    AB          01  90 00231    MOVB    #1, 90(CTX)                             ; 2405
                                        5E  DD 00235    PUSHL   SP                                      ; 2407
                  FDC4  CF              01  FB 00237    CALLS   #1, WORK_OPEN                           ;
                                 5A     AB  95 0023C    TSTB    90(CTX)                                 ; 2408
                                        0A  12 0023F    BNEQ    20$                                     ;
                     001C8014   8F      DD 00241        PUSHL   #1867796                                ; 2410
                            6A          01  FB 00247    CALLS   #1, SOR$$ERROR                          ;
                                        04  00024A      RET                                             ;
                            51    00D0  CB  9E 0024B 20$: MOVAB   208(CTX), R1                            ; 2421
                            50    0088  CB  3C 00250    MOVZWL  136(CTX), R0                            ; 2423
                            50    0200  C0  9E 00255    MOVAB   512(R0), R0                             ;
                            61          50  D1 0025A    CMPL    R0, (R1)                                ; 2424
                                        03  18 0025D    BGEQ    21$                                     ;
                            50          61  D0 0025F    MOVL    (R1), R0                                ;
                        00006400    8F      50  D1 00262 21$: CMPL    R0, #25600                      ; 2421
                                            05  18 00269    BGEQ    22$                                 ;
                                50      6400  8F  3C 0026B    MOVZWL  #25600, R0                      ;
                                61            50  D0 00270 22$: MOVL    R0, (R1)                        ;
                          50            61 000001FF   8F  C1 00273    ADDL3   #511, (R1), R0          ; 2425
                          61            50 000001FF   8F  CB 0027B    BICL3   #511, R0, (R1)          ;
                                                      04  00283 23$: RET                              ; 2427

; Routine Size:  644 bytes,    Routine Base:  SOR$RO_CODE + 09D9
```

```
;  2386          2428  1 GLOBAL ROUTINE SOR$$WRK_ALQ: CAL_CTXREG =
;  2387          2429  1
;  2388          2430  1 !++
;  2389          2431  1 !
;  2390          2432  1 ! FUNCTIONAL DESCRIPTION:
;  2391          2433  1 !
;  2392          2434  1 !       Calculate work file allocation.
;  2393          2435  1 !
;  2394          2436  1 ! FORMAL PARAMETERS:
;  2395          2437  1 !
;  2396          2438  1 !       NONE
;  2397          2439  1 !
;  2398          2440  1 ! IMPLICIT INPUTS:
;  2399          2441  1 !
;  2400          2442  1 !       NONE
;  2401          2443  1 !
;  2402          2444  1 ! IMPLICIT OUTPUTS:
;  2403          2445  1 !
;  2404          2446  1 !       NONE
;  2405          2447  1 !
;  2406          2448  1 ! ROUTINE VALUE:
;  2407          2449  1 !
;  2408          2450  1 !       Total work file allocation.
;  2409          2451  1 !
;  2410          2452  1 ! SIDE EFFECTS:
;  2411          2453  1 !
;  2412          2454  1 !       NONE
;  2413          2455  1 !
;  2414          2456  1 !--
;  2415          2457  2     BEGIN
;  2416          2458  2     EXTERNAL REGISTER
;  2417          2459  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  2418          2460  2     LOCAL
;  2419          2461  2         WFB:    REF WFB_BLOCK,
;  2420          2462  2         WFALQ:  INITIAL(0);
;  2421          2463  2
;  2422          2464  2     ! Loop through all the work files,
;  2423          2465  2     ! computing the total number of blocks allocated to them.
;  2424          2466  2     !
;  2425          2467  2     WFB = .CTX[S_WRK_WFB];
;  2426          2468  2     WHILE WFB[BASE_] NEQ 0 DO
;  2427          2469  2         (WFALQ = .WFALQ + .WFB[WFB_ALQ]; WFB = .WFB[WFB_NEXT]);
;  2428          2470  2     RETURN .WFALQ;
;  2429          2471  2
;  2430          2472  1     END;


                                  0000 00000          .ENTRY  SOR$$WRK_ALQ, Save nothing                      ;  2428
                              51  D4 00002            CLRL    WFALQ                                           ;  2457
                          50  00E4  CB  D0 00004        MOVL    228(CTX), WFB                                   ;  2467
                                   09  13 00009 1$:     BEQL    2$                                              ;  2468
                              51    0C  A0  C0 0000B    ADDL2   12(WFB), WFALQ                                  ;  2469
                              50        60  D0 0000F    MOVL    (WFB), WFB                                      ;
                                        F5  11 00012    BRB     1$                                              ;
```

```
                50          51  DO 00014 2$:    MOVL    WFALQ, R0                       ; 2470
                    04 00017    RET                                                     ; 2472

; Routine Size:  24 bytes,    Routine Base:  SOR$RO_CODE + 0C5D
```

```
;  2432          2473  1 ROUTINE CLEAN_UP: CAL_CTXREG NOVALUE =
;  2433          2474  1
;  2434          2475  1 !++
;  2435          2476  1 !
;  2436          2477  1 ! FUNCTIONAL DESCRIPTION:
;  2437          2478  1 !
;  2438          2479  1 !       Release resources allocated by this module.
;  2439          2480  1 !
;  2440          2481  1 ! FORMAL PARAMETERS:
;  2441          2482  1 !
;  2442          2483  1 !       NONE
;  2443          2484  1 !
;  2444          2485  1 ! IMPLICIT INPUTS:
;  2445          2486  1 !
;  2446          2487  1 !       NONE
;  2447          2488  1 !
;  2448          2489  1 ! IMPLICIT OUTPUTS:
;  2449          2490  1 !
;  2450          2491  1 !       NONE
;  2451          2492  1 !
;  2452          2493  1 ! ROUTINE VALUE:
;  2453          2494  1 !
;  2454          2495  1 !       NONE (signals errors)
;  2455          2496  1 !
;  2456          2497  1 ! SIDE EFFECTS:
;  2457          2498  1 !
;  2458          2499  1 !       NONE
;  2459          2500  1 !
;  2460          2501  1 !--
;  2461          2502  2     BEGIN
;  2462          2503  2     EXTERNAL REGISTER
;  2463          2504  2         CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
;  2464          2505  2     LOCAL
;  2465          2506  2         WFB:    REF WFB_BLOCK,
;  2466          2507  2         WFB_ADR:REF VECTOR[1],
;  2467          2508  2         STATUS;
;  2468          2509  2
;  2469          2510  2
;  2470          2511  2     ! Cancel the post-checkpoint handler
;  2471          2512  2     !
;  2472        U 2513  2     %IF FUN_K_CHECKPOINT %THEN
;  2473        U 2514  2     IF CHK$CANPSH NEQ 0 THEN
;  2474        U 2515  2     IF .CTX[S_PSH_HANDLER] NEQ 0
;  2475        U 2516  2     THEN
;  2476        U 2517  2         BEGIN
;  2477        U 2518  2         STATUS = CHK$CANPSH(CTX[S_PSH_HANDLER]);
;  2478        U 2519  2         IF .STATUS AND .CTX[S_RSH_HANDLER] NEQ 0 THEN
;  2479        U 2520  2         STATUS = CHK$CANRSH(CTX[S_RSH_HANDLER]);
;  2480        U 2521  2         IF NOT .STATUS
;  2481        U 2522  2         THEN
;  2482        U 2523  2             SOR$$ERROR(SOR$_SHR_SYSERROR AND NOT STS$M_SEVERITY OR STS$K_WARNING,
;  2483        U 2524  2                 0, .STATUS);
;  2484        U 2525  2         END;
;  2485          2526  2     %FI
;  2486          2527  2
;  2487          2528  2     ! Deallocate the work file event flag
;  2488          2529  2     !
```

```
;  2489          2530  2      %IF NOT HOSTILE %THEN
;  2490          2531  2      IF .CTX[S_WRK_EFN] NEQ 0
;  2491          2532  2      THEN
;  2492          2533  3          BEGIN
;  2493          2534  3          LOCAL
;  2494          2535  3              STATUS;
;  2495          2536  3          STATUS = LIB$FREE_EF(CTX[S_WRK_EFN]);
;  2496          2537  3          IF NOT .STATUS THEN SOR$$ERROR(
;  2497          2538  3                  SOR$_SHR_SYSERROR AND NOT STS$M_SEVERITY OR STS$K_ERROR,
;  2498          2539  3                  0, .STATUS);
;  2499          2540  2          END;
;  2500          2541  2      %FI
;  2501          2542  2
;  2502          2543  2
;  2503          2544  2      ! Deallocate the run description blocks from the active list.
;  2504          2545  2      !
;  2505          2546  3      BEGIN
;  2506          2547  3      LOCAL
;  2507          2548  3          Q:      REF VECTOR[1],
;  2508          2549  3          RUN:    REF RUN_BLOCK;
;  2509          2550  3      Q = CTX[S_RUN_INFO];
;  2510          2551  3      WHILE (RUN = .Q[0]) NEQ 0 DO DELRUN(RUN[BASE_]);
;  2511          2552  2      END;
;  2512          2553  2
;  2513          2554  2
;  2514          2555  2      ! For each WFB linked into this list
;  2515          2556  2      !
;  2516          2557  2      WFB_ADR = CTX[S_WRK_WFB];
;  2517          2558  2      WHILE (WFB = .WFB_ADR[0]) NEQ 0 DO
;  2518          2559  3          BEGIN
;  2519          2560  3
;  2520          2561  3          ! Deassign the channel
;  2521          2562  3          !
;  2522          2563  3          IF .WFB[WFB_CHAN] NEQ 0
;  2523          2564  3          THEN
;  2524          2565  4              BEGIN
;  2525        P 2566  4              STATUS = $DASSGN(CHAN=.WFB[WFB_CHAN]
;  2526          2567  4                  %IF HOSTILE_ELAN %THEN , E[AN = .CTX[COM_CTXADR] %FI );
;  2527          2568  4              IF NOT .STATUS THEN SOR$$ERROR(
;  2528          2569  4                  SOR$_SHR_SYSERROR AND NOT STS$M_SEVERITY OR STS$K_ERROR,
;  2529          2570  4                  0, .STATUS);
;  2530          2571  4              WFB[WFB_CHAN] = 0;
;  2531          2572  3              END;
;  2532          2573  3
;  2533          2574  3          ! Free the file name string
;  2534          2575  3          !
;  2535          2576  3          SOR$$FREE_FILE_NAME(WFB[WFB_NAME]);      ! Free name string
;  2536          2577  3
;  2537          2578  3          ! Free the RDBs linked to this WFB
;  2538          2579  3          !
;  2539          2580  4          BEGIN
;  2540          2581  4          LOCAL
;  2541          2582  4              Q:          REF VECTOR[1],
;  2542          2583  4              RDB:        REF RDB_BLOCK;
;  2543          2584  4          Q = WFB[WFB_FREE];
;  2544          2585  4          WHILE (RDB = .Q[0]) NEQ 0 DO REMLH_(Q[0], RDB, RDB_NEXT, RDB_K_SIZE);
;  2545          2586  4          Q = WFB[WFB_NOUSE];
```

```
;  2546          2587  4          WHILE (RDB = .Q[0]) NEQ 0 DO REMLH_(Q[0], RDB, RDB_NEXT, RDB_K_SIZE);
;  2547          2588  4          Q = WFB[WFB_FULL];
;  2548          2589  4          WHILE (RDB = .Q[0]) NEQ 0 DO REMLH_(Q[0], RDB, RDB_NEXT, RDB_K_SIZE);
;  2549          2590  3          END;
;  2550          2591  3
;  2551          2592  3          ! Free the WFB
;  2552          2593  3          !
;  2553          2594  3          REMLH_(WFB_ADR[0], WFB, WFB_NEXT, WFB_K_SIZE);
;  2554          2595  3
;  2555          2596  2          END;
;  2556          2597  2
;  2557          2598  2
;  2558          2599  2      ! Deallocate the buffers
;  2559          2600  2      !
;  2560          2601  3      BEGIN
;  2561          2602  3      LOCAL
;  2562          2603  3          Q:      REF VECTOR[1],
;  2563          2604  3          BUF:    REF BUF_BLOCK,
;  2564          2605  3          P:      REF BUF_BLOCK;
;  2565          2606  3      Q = CTX[S_BUF_ALLOC];
;  2566          2607  3      WHILE (P = .Q[0]) NEQ 0 DO
;  2567          2608  4          BEGIN
;  2568          2609  4          BUF = .P[BUF_ADDR];
;  2569          2610  4          BUF[BUF_NEXT] = .P[BUF_NEXT];
;  2570          2611  4          BUF[BUF_ADDR] = .P[BUF_ADDR];
;  2571          2612  4          Q[0] = BUF[BASE_];
;  2572          2613  4          REMLH_(Q[0], BUF, BUF_NEXT, .CTX[S_BUF_SIZE]+COM_K_BPERPAGE);
;  2573          2614  3          END;
;  2574          2615  2      END;
;  2575          2616  2
;  2576          2617  2
;  2577          2618  1      END;


                                    01FC 00000 CLEAN_UP:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8                ; 2473
                      58 00000000G   00  9E 00002       MOVAB   SOR$$ERROR, R8                   ;
                      57 00000000G   00  9E 00009       MOVAB   SOR$$DEALLOCATE, R7              ;
                           00E0      CB  D5 00010       TSTL    224(CTX)                         ; 2531
                                     1B  13 00014       BEQL    1$                               ;
                           00E0      CB  9F 00016       PUSHAB  224(CTX)                         ; 2536
                00000000G  00        01  FB 0001A       CALLS   #1, LIB$FREE_EF                  ;
                           0D        50  E8 00021       BLBS    STATUS, 1$                       ; 2537
                                     50  DD 00024       PUSHL   STATUS                           ; 2539
                                     7E  D4 00026       CLRL    -(SP)                            ; 2537
                         001C11B2    8F  DD 00028       PUSHL   #1839538                         ; 2538
                           68        03  FB 0002E       CALLS   #3, SOR$$ERROR                   ;
                           53   00D8 CB  9E 00031 1$:    MOVAB   216(R11), Q                      ; 2550
                           52        63  D0 00036 2$:    MOVL    (Q), RUN                         ; 2551
                                     09  13 00039       BEQL    3$                               ;
                                     52  DD 0003B       PUSHL   RUN                              ;
                              FC8A   CF  01  FB 0003D       CALLS   #1, DELRUN                   ;
                                     F2  11 00042       BRB     2$                               ;
                           55   00E4 CB  9E 00044 3$:    MOVAB   228(R11), WFB_ADR                ; 2557
```

```
                52              65 D0 00049 4$:     MOVL    (WFB_ADR), WFB                  ; 2558
                                77 13 0004C         BEQL    13$
                      1E        A2 B5 0004E         TSTW    30(WFB)                         ; 2563
                                21 13 00051         BEQL    6$
                7E      1E        A2 3C 00053         MOVZWL  30(WFB), -(SP)                  ; 2567
00000000G       00                01 FB 00057         CALLS   #1, SYS$DASSGN
                56                50 D0 0005E         MOVL    R0, STATUS
                0D                56 E8 00061         BLBS    STATUS, 5$                      ; 2568
                                  56 DD 00064         PUSHL   STATUS                          ; 2570
                                  7E D4 00066         CLRL    -(SP)                           ; 2568
                   001C11B2       8F DD 00068         PUSHL   #1839538                        ; 2569
                68                03 FB 0006E         CALLS   #3, SOR$$ERROR
                      1E        A2 B4 00071 5$:     CLRW    30(WFB)                         ; 2571
                      04        A2 9F 00074 6$:     PUSHAB  4(WFB)                          ; 2576
00000000G       00                01 FB 00077         CALLS   #1, SOR$$FREE_FILE_NAME
                53      10        A2 9E 0007E         MOVAB   16(R2), Q                       ; 2584
                54                63 D0 00082 7$:     MOVL    (Q), RDB                        ; 2585
                                  0B 13 00085         BEQL    8$
                                  64 DD 00087         PUSHL   (RDB)
                                  53 DD 00089         PUSHL   Q
                                  10 DD 0008B         PUSHL   #16
                67                03 FB 0008D         CALLS   #3, SOR$$DEALLOCATE
                                  F0 11 00090         BRB     7$
                53      14        A2 9E 00092 8$:     MOVAB   20(R2), Q                       ; 2586
                54                63 D0 00096 9$:     MOVL    (Q), RDB                        ; 2587
                                  0B 13 00099         BEQL    10$
                                  64 DD 0009B         PUSHL   (RDB)
                                  53 DD 0009D         PUSHL   Q
                                  10 DD 0009F         PUSHL   #16
                67                03 FB 000A1         CALLS   #3, SOR$$DEALLOCATE
                                  F0 11 000A4         BRB     9$
                53        18      A2 9E 000A6 10$:    MOVAB   24(R2), Q                       ; 2588
                54                63 D0 000AA 11$:    MOVL    (Q), RDB                        ; 2589
                                  0B 13 000AD         BEQL    12$
                                  64 DD 000AF         PUSHL   (RDB)
                                  53 DD 000B1         PUSHL   Q
                                  10 DD 000B3         PUSHL   #16
                67                03 FB 000B5         CALLS   #3, SOR$$DEALLOCATE
                                  F0 11 000B8         BRB     11$
                                  62 DD 000BA 12$:    PUSHL   (WFB)                           ; 2594
                                  55 DD 000BC         PUSHL   WFB_ADR
                                  20 DD 000BE         PUSHL   #32
                67                03 FB 000C0         CALLS   #3, SOR$$DEALLOCATE
                                  84 11 000C3         BRB     4$                              ; 2558
                54      00E8      CB 9E 000C5 13$:    MOVAB   232(R11), Q                     ; 2606
                52                64 D0 000CA 14$:    MOVL    (Q), P                          ; 2607
                                  1D 13 000CD         BEQL    15$
                53        04      A2 D0 000CF         MOVL    4(P), BUF                       ; 2609
                63                62 7D 000D3         MOVQ    (P), (BUF)                      ; 2610
                64                53 D0 000D6         MOVL    BUF, (Q)                        ; 2612
                                  63 DD 000D9         PUSHL   (BUF)                           ; 2613
                                  54 DD 000DB         PUSHL   Q
7E      00D0      CB 00000200     8F C1 000DD         ADDL3   #512, 208(CTX), -(SP)
                67                03 FB 000E7         CALLS   #3, SOR$$DEALLOCATE
                                  DE 11 000EA         BRB     14$                             ; 2607
                                     04 000EC 15$:    RET                                     ; 2618
```

; Routine Size: 237 bytes,   Routine Base: SOR$RO_CODE + 0C75

```
; 2579          2619  1 END
; 2580          2620  0 ELUDOM
```

```
;                                  PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  SOR$RO_CODE--------------------2      4  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  SOR$RO_CODE                        3426  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       131         1         581       00:01.0
;  _$255$DUA28:[SYSLIB]XPORT.L32;1      590        29         4         252       00:00.6
;  _$255$DUA28:[SORT32.SRC]SORLIB.L32;1 409       150        36          34       00:00.4
;  _$255$DUA28:[SORT32.SRC]SRTSPC.L32;1 120         3         2          12       00:00.1
```

```
;                                  COMMAND QUALIFIERS

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

; Size:          3366 code + 64 data bytes
; Run Time:         01:23.8
; Elapsed Time:     04:29.6
; Lines/CPU Min:     1876
; Lexemes/CPU-Min: 42112
; Memory Used:  294 pages
; Compilation Complete
```

0365 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SORRMSIO
LIS

SORMSG
LIS

SORSCRIO
LIS

SOROUTPUT
LIS

SORLIB
LIS

0366 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SRTSPCBLI
LIS

SORSPEC
LIS

SORTRNVEC
LIS

SORSORT
LIS

SORSPCUTI
LIS

SRTSPC
LIS