```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFFFFFFFFFF       000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFF       000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFF       000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
```

FILEID**FORRECPRO

```
FFFFFFFFFF    000000    RRRRRRRR    RRRRRRRR    EEEEEEEEEE    CCCCCCCC  PPPPPPPP    RRRRRRRR      000000
FFFFFFFFFF    000000    RRRRRRRR    RRRRRRRR    EEEEEEEEEE    CCCCCCCC  PPPPPPPP    RRRRRRRR      000000
FF          00      00  RR      RR  RR      RR  EE          CC          PP      PP  RR      RR  00      00
FF          00      00  RR      RR  RR      RR  EE          CC          PP      PP  RR      RR  00      00
FF          00      00  RR      RR  RR      RR  EE          CC          PP      PP  RR      RR  00      00
FF          00      00  RR      RR  RR      RR  EE          CC          PP      PP  RR      RR  00      00
FFFFFFFF    00      00  RRRRRRRR    RRRRRRRR    EEEEEEEE    CC          PPPPPPPP    RRRRRRRR    00      00
FFFFFFFF    00      00  RRRRRRRR    RRRRRRRR    EEEEEEEE    CC          PPPPPPPP    RRRRRRRR    00      00
FF          00      00  RR  RR      RR  RR      EE          CC          PP          RR  RR      00      00
FF          00      00  RR  RR      RR  RR      EE          CC          PP          RR  RR      00      00
FF          00      00  RR    RR    RR    RR    EE          CC          PP          RR    RR    00      00
FF          00      00  RR    RR    RR    RR    EE          CC          PP          RR    RR    00      00      ....
FF            000000    RR      RR  RR      RR  EEEEEEEEEE    CCCCCCCC  PP          RR      RR    000000      ....
FF            000000    RR      RR  RR      RR  EEEEEEEEEE    CCCCCCCC  PP          RR      RR    000000      ....

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

```
;     1          0001  0 MODULE FOR$$REC_PROC (%TITLE'Record processing level of abstraction'
;     2          0002  0                 IDENT = '1-031'         ! file: FORRECPRO.B32  Edit: SBL1031
;     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 ! FACILITY: FORTRAN Support Library - not user callable
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This module implements the record processing level of
;    35          0035  1 !       abstraction which is the 3rd level and is called only from
;    36          0036  1 !       the user data formatter level (2nd level) when the user
;    37          0037  1 !       portion of a record buffer is full (WRITE) or empty
;    38          0038  1 !       (READ). This module adds any per record formatting (as
;    39          0039  1 !       distinguished from per I/O statement or per I/O list element
;    40          0040  1 !       formatting) and then calls RMS ($PUT or $GET). RMS errors
;    41          0041  1 !       are converted to FORTRAN errors and are signaled.
;    42          0042  1 !
;    43          0043  1 ! ENVIRONMENT:  User access mode; AST level or not.
;    44          0044  1 !
;    45          0045  1 ! AUTHOR:       Thomas N. Hastings;     CREATION DATE: 16-Mar-77
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !         Previous edit history deleted.  SBL 18-July-1980
;    50          0050  1 ! 1-001 - Update version number and copyright notice.  JBS 16-NOV-78
;    51          0051  1 ! 1-002 - Change REQUIRE file names from FOR... to OTS...  JBS 06-DEC-78
;    52          0052  1 ! 1-003 - Change ISB$A_BUF_PTR, BUF_BEG, BUF_END, BUF_HIGH to LUB. DGP 08-Jan-79
;    53          0053  1 ! 1-004 - Always set RAB$W_RSZ before $PUT.  SPR 22044 SBL 31-Jan-79
;    54          0054  1 ! 1-005 - For sequential read, set access mode to SEQ.  For sequential
;    55          0055  1 !         write, set to KEY if ACCESS='KEYED', SEQ otherwise.  SBL 10-Apr-79
;    56          0056  1 ! 1-006 - Change dispatch table so that direct "1" level is the same as
;    57          0057  1 !         "0" level.  This is because we now allow more than one record
```

```
;    58          0058  1 !          on direct I/O.  SBL 25-Apr-1979
;    59          0059  1 ! 1-007 - Clear RAB$V_UIF if sequential or keyed write.  SBL 2-May-1979
;    60          0060  1 ! 1-008 - Add REC level definitions for keyed read.  SBL 2-May-1979
;    61          0061  1 ! 1-009 - Change FOR$K_TARRECLOC to FOR$K_SPERECLOC.  SBL 8-May-1979
;    62          0062  1 ! 1-010 - Add REWRITE.  SBL 14-May-1979
;    63          0063  1 ! 1-011 - Clear LUB$V_FIND_LAST on direct access.  SBL 15-May-1979
;    64          0064  1 ! 1-012 - Add internal file entry points.
;    65          0065  1 ! 1-013 - Remove initial space from REC_WSL, transferred to UDF level.
;    66          0066  1 !          Make list directed record length 80 instead of 72.  SBL 26-Jun-1979
;    67          0067  1 ! 1-014 - Check for ISB$V_LAST_REC set in FOR$$REC_RSU1 to catch reading
;    68          0068  1 !          too much data.  SPR 22-25598  SBL 17-Aug-1979
;    69          0069  1 ! 1-015 - Change FOR$K_MORONEREC to FOR$K_TOOMANREC.  SBL 21-Sept-1979
;    70          0070  1 ! 1-016 - Change FOR$K_ATTREANON to FOR$K_ATTACCNON.  JBS 24-SEP-1979
;    71          0071  1 ! 1-017 - Use LUB$W_R_MARGIN for list directed output width.  SBL 4-Oct-1979
;    72          0072  1 ! 1-018 - Give FOR$_SPERECLOC on read errors.  SBL 12-Oct-1979
;    73          0073  1 ! 1-019 - Use LUB$W_R_MARGIN as list directed output record length.  SBL 1-Nov-1979
;    74          0074  1 ! ***** - VAX/VMS V2.0
;    75          0075  1 ! 1-020 - Add support for NAMELIST.  SBL 18-July-1980
;    76          0076  1 ! 1-021 - If fixed length records, blank fill list directed output records.
;    77          0077  1 !          SBL 27-August-1980
;    78          0078  1 ! 1-022 - Create separate routines FOR$$REC_WSN0 and FOR$$REC_WSN1 for
;    79          0079  1 !          NAMELIST.  They require unique code not found in WSFx.  SBL 4-September-1980
;    80          0080  1 ! 1-023 - Only check for ENDFILE record on segmented or formatted sequential
;    81          0081  1 !          organization/access files.  SBL 29-Sept-1980
;    82          0082  1 ! 1-024 - Don't write leading space in FOR$$REC_WSN0.  The UDF routine will do it.  SBL 21-Oct-1980
;    83          0083  1 ! 1-025 - Define FOR$$REC_WSL9 as FOR$$REC_WSL1 rather than FOR$$REC_WSF1.  This
;    84          0084  1 !          is so that edit 1-021 works for end-of-statement.  SBL 5-Nov-1980
;    85          0085  1 ! 1-026 - Make sure ENDFILE record gets counted before signaling FOR$K_ENDDURREA
;    86          0086  1 !          in FOR$$REC_RSF0, so backspace will work correctly.  JAW 24-Feb-1981
;    87          0087  1 ! 1-027 - Implement unbuffered transfers for single-element I/O lists:
;    88          0088  1 !          If ISB$V_SNGL_ELEM is set, do not change RAB$L_RBF and
;    89          0089  1 !          RAB$W_RSZ, which have been set at UDF level.  JAW 06-May-1981
;    90          0090  1 ! 1-028 - Continuation of 1-027.  Support recordtype=variable.  JAW
;    91          0091  1 !          02-Jun-1981
;    92          0092  1 ! 1-029 - Check for fixed-length records in FOR$$REC_WD9 before padding
;    93          0093  1 !          them.  Also, be sure first byte, or first two bytes, of record
;    94          0094  1 !          read in FOR$$REC_RSF0 and FOR$$REC_RSU0 exist before testing
;    95          0095  1 !          their contents.  JAW 06-Jun-1981
;    96          0096  1 ! ***** - VAX/VMS V3.0
;    97          0097  1 ! 1-030 - Add entries for list-directed internal files.  Use prologue file.
;    98          0098  1 !          SBL 21-Apr-1983
;    99          0099  1 ! 1-031 - Add extra space in FOR$$REC_WSN0 if statement type is a READ.  This
;   100          0100  1 !          makes the NAMELIST '?' feature display with a leading blank.
;   101          0101  1 !          SBL 26-May-1983
;   102          0102  1 !--
```

```
;  104          0103  1 !
;  105          0104  1 ! PROLOGUE FILE:
;  106          0105  1 !
;  107          0106  1
;  108          0107  1 REQUIRE 'RTLIN:FORPROLOG';                      ! FORTRAN definitions
;  109          0173  1
;  110          0174  1 !
;  111          0175  1 ! TABLE OF CONTENTS:
;  112          0176  1 !
;  113          0177  1 ! write sequential formatted:
;  114          0178  1
;  115          0179  1 FORWARD ROUTINE
;  116          0180  1     FOR$$REC_WSF0 : JSB_REC0 NOVALUE,           ! initialize output buffer
;  117          0181  1     FOR$$REC_WSF1 : JSB_REC1 NOVALUE;           ! write all but last record
;  118          0182  1
;  119          0183  1 GLOBAL BIND
;  120          0184  1     ROUTINE
;  121          0185  1     FOR$$REC_WSF9 = FOR$$REC_WSF1;              ! Write next = write last
;  122          0186  1
;  123          0187  1 ! read sequential formatted:
;  124          0188  1
;  125          0189  1 FORWARD ROUTINE
;  126          0190  1     FOR$$REC_RSF0 : JSB_REC0 NOVALUE;           ! read first record
;  127          0191  1
;  128          0192  1 GLOBAL BIND
;  129          0193  1     ROUTINE
;  130          0194  1     FOR$$REC_RSF1 = FOR$$REC_RSF0;              ! Read first = read next
;  131          0195  1
;  132          0196  1 FORWARD ROUTINE
;  133          0197  1     FOR$$REC_RSF9 : JSB_REC9 NOVALUE,           ! no-op
;  134          0198  1     ! read sequential unformatted record
;  135          0199  1     FOR$$REC_RSU0 : JSB_REC0 NOVALUE,           ! read first record
;  136          0200  1     FOR$$REC_RSU1 : JSB_REC1 NOVALUE,           ! read all subsequent records
;  137          0201  1     FOR$$REC_RSU9 : JSB_REC9 NOVALUE,           ! terminate read
;  138          0202  1     ! write sequential unformatted record:
;  139          0203  1     FOR$$REC_WSU0 : JSB_REC0 NOVALUE,           ! initialize output buffer
;  140          0204  1     FOR$$REC_WSU1 : JSB_REC1 NOVALUE,           ! write all but last record
;  141          0205  1     FOR$$REC_WSU9 : JSB_REC9 NOVALUE,           ! write last record
;  142          0206  1     ! write direct (formatted and unformatted):
;  143          0207  1     FOR$$REC_WD0 : JSB_REC0 NOVALUE,            ! initialize output buffer
;  144          0208  1     FOR$$REC_WD1 : JSB_REC1 NOVALUE,            ! write next record
;  145          0209  1     FOR$$REC_WD9 : JSB_REC9 NOVALUE,            ! write last record
;  146          0210  1     ! read direct (formatted and unformatted):
;  147          0211  1     FOR$$REC_RD0 : JSB_REC0 NOVALUE;            ! read first record
;  148          0212  1
;  149          0213  1 GLOBAL BIND
;  150          0214  1     ROUTINE
;  151          0215  1     FOR$$REC_RD1 = FOR$$REC_RD0,                ! next record = first record
;  152          0216  1     FOR$$REC_RD9 = FOR$$REC_RSF9;               ! no-op
;  153          0217  1
;  154          0218  1 ! write sequential list-directed
;  155          0219  1
;  156          0220  1 FORWARD ROUTINE
;  157          0221  1     FOR$$REC_WSL0 : JSB_REC0 NOVALUE,           ! initialize output buffer
;  158          0222  1     FOR$$REC_WSL1 : JSB_REC1 NOVALUE;           ! write all but last record
;  159          0223  1
;  160          0224  1 GLOBAL BIND
```

```
;  161          0225  1      ROUTINE
;  162          0226  1      FOR$$REC_WSL9 = FOR$$REC_WSF1,          ! write last = write last formatted sequential
;  163          0227  1      ! read sequential list-directed
;  164          0228  1      FOR$$REC_RSL0 = FOR$$REC_RSF0,          ! read first = read first sequential formatted
;  165          0229  1      FOR$$REC_RSL1 = FOR$$REC_RSF1,          ! read next = read next sequential formatted
;  166          0230  1      FOR$$REC_RSL9 = FOR$$REC_RSF9,          ! no-op
;  167          0231  1      ! read memory formatted (DECODE)
;  168          0232  1      FOR$$REC_RMF0 = FOR$$REC_RSF9;          ! no-op
;  169          0233  1
;  170          0234  1  FORWARD ROUTINE
;  171          0235  1      FOR$$REC_RMF1 : JSB_REC1 NOVALUE;       ! illegal: FOR$_TOOMANREC
;  172          0236  1
;  173          0237  1  GLOBAL BIND
;  174          0238  1      ROUTINE
;  175          0239  1      FOR$$REC_RMF9 = FOR$$REC_RSF9,          ! no-op
;  176          0240  1      ! write memory formatted (ENCODE)
;  177          0241  1      FOR$$REC_WMF0 = FOR$$REC_RSF9,          ! no-op
;  178          0242  1      FOR$$REC_WMF1 = FOR$$REC_RMF1;          ! illegal: FOR$_TOOMANREC
;  179          0243  1
;  180          0244  1  FORWARD ROUTINE
;  181          0245  1      FOR$$REC_WMF9 : JSB_REC9 NOVALUE,       ! terminate write (blank pad)
;  182          0246  1      FOR$$REC_WIF0 : JSB_REC0 NOVALUE,       ! Internal file write
;  183          0247  1      FOR$$REC_WIF1 : JSB_REC1 NOVALUE;       !
;  184          0248  1
;  185          0249  1  GLOBAL BIND
;  186          0250  1      ROUTINE
;  187          0251  1      FOR$$REC_WIF9 = FOR$$REC_WMF9,          ! Terminate internal write
;  188          0252  1      FOR$$REC_RIF0 = FOR$$REC_WIF0,          ! Internal file read
;  189          0253  1      FOR$$REC_RIF1 = FOR$$REC_WIF1,          !
;  190          0254  1      FOR$$REC_RIF9 = FOR$$REC_RSF9;          ! No-op
;  191          0255  1
;  192          0256  1  FORWARD ROUTINE
;  193          0257  1      FOR$$REC_RKF0 : JSB_REC0 NOVALUE;       ! keyed read
;  194          0258  1
;  195          0259  1  GLOBAL BIND
;  196          0260  1      ROUTINE
;  197          0261  1      FOR$$REC_RKF1 = FOR$$REC_RSF0,          ! 2nd and after like sequential
;  198          0262  1      FOR$$REC_RKF9 = FOR$$REC_RSF9,          ! No-op
;  199          0263  1      FOR$$REC_RKU0 = FOR$$REC_RKF0,          ! Keyed unformatted read
;  200          0264  1      FOR$$REC_RKU1 = FOR$$REC_RSU1,          ! Will be an error
;  201          0265  1      FOR$$REC_RKU9 = FOR$$REC_RSF9,          ! No-op
;  202          0266  1      FOR$$REC_WXF0 = FOR$$REC_WSF0,          ! Indexed rewrite
;  203          0267  1      FOR$$REC_WXF1 = FOR$$REC_RMF1;          ! Error
;  204          0268  1
;  205          0269  1  FORWARD ROUTINE
;  206          0270  1      FOR$$REC_WXF9 : JSB_REC9 NOVALUE;       ! Rewrite the record
;  207          0271  1
;  208          0272  1  GLOBAL BIND
;  209          0273  1      ROUTINE
;  210          0274  1      FOR$$REC_WXU0 = FOR$$REC_WSU0,          ! Rewrite indexed unformatted
;  211          0275  1      FOR$$REC_WXU1 = FOR$$REC_RMF1,          ! Error
;  212          0276  1      FOR$$REC_WXU9 = FOR$$REC_WXF9;          !
;  213          0277  1
;  214          0278  1  FORWARD ROUTINE
;  215          0279  1      FOR$$REC_WSN0 : JSB_REC0 NOVALUE,       ! Write sequential NAMELIST
;  216          0280  1      FOR$$REC_WSN1 : JSB_REC1 NOVALUE;
;  217          0281  1
```

```
;  218          0282  1 GLOBAL BIND
;  219          0283  1     FOR$$REC_RSN0 = FOR$$REC_RSF0,          ! Read sequential NAMELIST
;  220          0284  1     FOR$$REC_RSN1 = FOR$$REC_RSF1,
;  221          0285  1     FOR$$REC_WIL0 = FOR$$REC_WIF0,          . Write internal list-directed
;  222          0286  1     FOR$$REC_WIL1 = FOR$$REC_WIF1,
;  223          0287  1     FOR$$REC_WIL9 = FOR$$REC_WMF9,
;  224          0288  1     FOR$$REC_RIL0 = FOR$$REC_RIF0,          ! Read internal list-directed
;  225          0289  1     FOR$$REC_RIL1 = FOR$$REC_RIF1,
;  226          0290  1     FOR$$REC_RIL9 = FOR$$REC_RSF9;          ! no-op
;  227          0291  1
;  228          0292  1 FORWARD ROUTINE
;  229          0293  1     FILL_BUF : CALL_CCB NOVALUE,            ! fill remainder of buffer with arg
;  230          0294  1     PUT_ERROR : CALL_CCB NOVALUE,           ! error in $PUT
;  231          0295  1     GET_ERROR : CALL_CCB NOVALUE;           ! error in $GET
;  232          0296  1
;  233          0297  1 !
;  234          0298  1 ! MACROS:
;  235          0299  1 !
;  236          0300  1 !       NONE
;  237          0301  1 !
;  238          0302  1 ! EQUATED SYMBOLS:
;  239          0303  1 !
;  240          0304  1 !       NONE
;  241          0305  1 !
;  242          0306  1 ! OWN STORAGE:
;  243          0307  1 !
;  244          0308  1 !       NONE
;  245          0309  1 !
;  246          0310  1 ! EXTERNAL REFERENCES:
;  247          0311  1 !
;  248          0312  1
;  249          0313  1 EXTERNAL ROUTINE
;  250          0314  1     FOR$$SIGNAL_STO : NOVALUE,              ! Convert FORTRAN err # to VAX
;  251          0315  1                                             ! err # and call LIB$STOP.
;  252          0316  1     FOR$$ASSOC : CALL_CCB NOVALUE;          ! store rec# in ass. var.
;  253          0317  1
```

```
;  255          0318  1 GLOBAL ROUTINE FOR$$REC_WSFO                    ! Write sequential formatted
;  256          0319  1     : JSB_REC0 NOVALUE =
;  257          0320  1
;  258          0321  1 !++
;  259          0322  1 ! FUNCTIONAL DESCRIPTION:
;  260          0323  1 !
;  261          0324  1 !       Initialize call for write sequential formatted.  FOR$$REC_WSF0
;  262          0325  1 !       initializes the output buffer and returns start and end+1 of user
;  263          0326  1 !       part of record buffer to be filled by caller.
;  264          0327  1 !
;  265          0328  1 ! CALLING SEQUENCE:
;  266          0329  1 !
;  267          0330  1 !       JSB FOR$$REC_WSF0 ()
;  268          0331  1 !
;  269          0332  1 ! FORMAL PARAMETERS:
;  270          0333  1 !       NONE
;  271          0334  1 !
;  272          0335  1 ! IMPLICIT INPUTS:
;  273          0336  1 !
;  274          0337  1 !       CCB                     Pointer to current logical unit
;  275          0338  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  276          0339  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
;  277          0340  1 !
;  278          0341  1 ! IMPLICIT OUTPUTS:
;  279          0342  1 !
;  280          0343  1 !       LUB$A_BUF_PTR           points to first char in user part of record
;  281          0344  1 !                               buffer
;  282          0345  1 !       LUB$A_BUF_END           points to last char+1 in user part
;  283          0346  1 !                               of record buffer
;  284          0347  1 !
;  285          0348  1 !       RAB$B_RAC               Set to correct access mode.
;  286          0349  1 !       RAB$V_UIF               Cleared if keyed access.
;  287          0350  1 !
;  288          0351  1 ! ROUTINE VALUE:
;  289          0352  1 !
;  290          0353  1 !       NONE
;  291          0354  1 !
;  292          0355  1 ! SIDE EFFECTS:
;  293          0356  1 !
;  294          0357  1 !       NONE
;  295          0358  1 !--
;  296          0359  1
;  297          0360  2     BEGIN
;  298          0361  2
;  299          0362  2     EXTERNAL REGISTER
;  300          0363  2         CCB : REF $FOR$CCB_DECL;
;  301          0364  2
;  302          0365  2     !+
;  303          0366  2     ! Return start address and end+1 address of output buffer.
;  304          0367  2     !-
;  305          0368  2
;  306          0369  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;  307          0370  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  308          0371  2
;  309          0372  2     !+
;  310          0373  2     ! If user opened with ACCESS='KEYED', then set access mode
;  311          0374  2     ! to KEYED and clear RAB$V_UIF else set access to sequential.
```

```
;  312          0375  2      !-
;  313          0376  2
;  314          0377  2      IF .CCB [LUB$V_KEYED]
;  315          0378  2      THEN
;  316          0379  3          BEGIN
;  317          0380  3          CCB [RAB$B_RAC] = RAB$C_KEY;
;  318          0381  3          CCB [RAB$V_UIF] = 0;
;  319          0382  3          END
;  320          0383  2      ELSE
;  321          0384  2          CCB [RAB$B_RAC] = RAB$C_SEQ;
;  322          0385  2
;  323          0386  2      RETURN;
;  324          0387  1      END;                                    ! END OF ROUTINE


;                                                                    .TITLE  FOR$$REC_PROC Record processing level of abstra
                                                                                 ction
                                                                     .IDENT  \1-031\

                                                                     .EXTRN  FOR$$SIGNAL_STO
                                                                     .EXTRN  FOR$$ASSOC

                                                                     .PSECT  _FOR$CODE,NOWRT,  SHR,  PIC,2

                    B0   AB        EC   AB  D0 00000 FOR$$REC_WSFO::
                                                     MOVL    -20(CCB), -80(CCB)                          ; 0369
                         50        D2   AB  3C 00005              MOVZWL  -46(CCB), R0                    ; 0370
                    B4   AB        EC BB40  9E 00009              MOVAB   @-20(CCB)[R0], -76(CCB)
                                   FD   AB  95 0000F              TSTB    -3(CCB)                         ; 0377
                                        09  18 00012              BGEQ    1$
                    1E   AB             01  90 00014              MOVB    #1, 30(CCB)                     ; 0380
                    04   AB             10  8A 00018              BICB2   #16, 4(CCB)                     ; 0381
                                        05  0001C                 RSB                                     ; 0377
                                   1E   AB  94 0001D 1$:          CLRB    30(CCB)                         ; 0384
                                        05  00020                 RSB                                     ; 0387

; Routine Size:  33 bytes,    Routine Base:  _FOR$CODE + 0000

;  325          0388  1
```

```
;  327     0389  1 GLOBAL ROUTINE FOR$$REC_WSF1                    ! Write sequential formatted
;  328     0390  1     ! (also FOR$$REC_WSF9 and FOR$$REC_WSL9)
;  329     0391  1     ! Called from FOR$$REC_WSU9
;  330     0392  1     : JSB_REC1 NOVALUE =
;  331     0393  1
;  332     0394  1 !++
;  333     0395  1 ! FUNCTIONAL DESCRIPTION:
;  334     0396  1 !
;  335     0397  1 !       Write one sequential formatted record and initialize for the next
;  336     0398  1 !       FOR$$REC_WSF1 (and FOR$$REC_WSF9) writes one output buffer and then
;  337     0399  1 !       initializes the output buffer and returns start and end+1 of user
;  338     0400  1 !       part of record buffer to be filled by caller.
;  339     0401  1 !       FLR records are space or null padded.
;  340     0402  1 !       Logical record number is incremented.
;  341     0403  1 !
;  342     0404  1 ! CALLING SEQUENCE:
;  343     0405  1 !
;  344     0406  1 !       JSB FOR$$REC_WSF1 ()
;  345     0407  1 !
;  346     0408  1 ! FORMAL PARAMETERS:
;  347     0409  1 !
;  348     0410  1 !       NONE
;  349     0411  1 !
;  350     0412  1 ! IMPLICIT INPUTS:
;  351     0413  1 !
;  352     0414  1 !       CCB                     Pointer to current logical unit
;  353     0415  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  354     0416  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
;  355     0417  1 !       LUB$A_BUF_END           points to last char inserted into buffer
;  356     0418  1 !                               by UDF level I/O.
;  357     0419  1 !       ISB$V_SNGL_ELEM         Flag indicating that RAB$W_RSZ and
;  358     0420  1 !                               RAB$L_RBF have been set up at UDF level
;  359     0421  1 !
;  360     0422  1 ! IMPLICIT OUTPUTS:
;  361     0423  1 !
;  362     0424  1 !       LUB$L_LOG_RECNO         Incremented logical record number
;  363     0425  1 !       LUB$A_BUF_PTR           Address of first char in user part
;  364     0426  1 !                               of record buffer
;  365     0427  1 !       LUB$A_BUF_END           Address of last+1 char in user part
;  366     0428  1 !                               of record buffer
;  367     0429  1 !
;  368     0430  1 ! ROUTINE VALUE:
;  369     0431  1 !
;  370     0432  1 !       NONE
;  371     0433  1 !
;  372     0434  1 ! SIDE EFFECTS:
;  373     0435  1 !
;  374     0436  1 !       NONE
;  375     0437  1 !--
;  376     0438  1
;  377     0439  2     BEGIN
;  378     0440  2
;  379     0441  2     EXTERNAL REGISTER
;  380     0442  2         CCB : REF $FOR$CCB_DECL;
;  381     0443  2
;  382     0444  2     !+
;  383     0445  2     ! If this is a buffered transfer (ISB$V_SNGL_ELEM = 0), then set up
```

```
;  384          0446  2      ! RAB$W_RSZ and RAB$L_RBF to point to the record buffer.  Otherwise
;  385          0447  2      ! they have already been set up at UDF level to point to an element.
;  386          0448  2      !-
;  387          0449  2
;  388          0450  2      IF NOT .CCB [ISB$V_SNGL_ELEM]
;  389          0451  2      THEN
;  390          0452  3          BEGIN
;  391          0453  3
;  392          0454  3          !+
;  393          0455  3          ! If fixed length records (FLR), pad with trailing spaces or
;  394          0456  3          ! nulls, depending on whether formatted or unformatted.  Set
;  395          0457  3          ! recordsize to actual length of record.
;  396          0458  3          !-
;  397          0459  3
;  398          0460  3          IF .CCB [LUB$V_FIXED]
;  399          0461  3          THEN
;  400          0462  4              BEGIN
;  401          0463  4              CCB [RAB$W_RSZ] = .CCB [LUB$W_RBUF_SIZE];   ! Always set RSZ
;  402          0464  4
;  403          0465  4              IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END]
;  404          0466  4              THEN
;  405          0467  4                  FILL_BUF (
;  406          0468  4
;  407          0469  4                      IF .CCB [LUB$V_UNFORMAT] THEN 0 ELSE %C' ')
;  408          0470  4
;  409          0471  4              END
;  410          0472  3          ELSE
;  411          0473  3              CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_RBUF_ADR];
;  412          0474  3          CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
;  413          0475  2          END;
;  414          0476  2
;  415          0477  2      !+
;  416          0478  2      ! Output buffer to RMS and check for errors
;  417          0479  2      ! If errors, SIGNAL_STO FOR$_ERRDURWRI (38='ERROR DURING WRITE')
;  418          0480  2      !-
;  419          0481  2
;  420          0482  2      IF NOT $PUT (RAB = .CCB) THEN PUT_ERROR ();
;  421          0483  2
;  422          0484  2      !+
;  423          0485  2      ! increment logical record number (for backspace) after
;  424          0486  2      ! successfully writing current record number.
;  425          0487  2      !-
;  426          0488  2
;  427          0489  2      IF NOT .CCB [LUB$V_KEYED] THEN CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
;  428          0490  2
;  429          0491  2      !+
;  430          0492  2      ! Return next output buffer start and end addresses
;  431          0493  2      ! If fixed length, use just record size, else buffer size.
;  432          0494  2      !-
;  433          0495  2
;  434          0496  2      CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;  435          0497  2      CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  436          0498  2      RETURN;
;  437          0499  1      END;                                            ! END OF ROUTINE
```

```
                                                        .EXTRN  SYS$PUT

                 2F      97   AB              04  E0 00000 FOR$$REC_WSF1::
                                                            BBS     #4, -105(CCB), 5$                       ; 0450
                 1E      FD   AB              02  E1 00005         BBC     #2, -3(CCB), 3$                 ; 0460
                         22   AB        D2    AB  B0 0000A         MOVW    -46(CCB), 34(CCB)               ; 0463
                         B4   AB        B0    AB  D1 0000F         CMPL    -80(CCB), -76(CCB)              ; 0465
                                              19  1E 00014         BGEQU   4$
                 04      FD   AB              01  E1 00016         BBC     #1, -3(CCB), 1$                 ; 0469
                                              7E  D4 0001B         CLRL    -(SP)
                                              02  11 0001D         BRB     2$
                                              20  DD 0001F 1$:     PUSHL   #32
                       0000V  CF              01  FB 00021 2$:     CALLS   #1, FILL_BUF
                                              07  11 00026         BRB     4$                              ; 0465
           22    AB       B0   AB        EC    AB  A3 00028 3$:     SUBW3   -20(CCB), -80(CCB), 34(CCB)     ; 0473
                          28   AB        EC    AB  D0 0002F 4$:     MOVL    -20(CCB), 40(CCB)               ; 0474
                                              5B  DD 00034 5$:     PUSHL   CCB                             ; 0482
                00000000G  00                 01  FB 00036         CALLS   #1, SYS$PUT
                           05                 50  E8 0003D         BLBS    R0, 6$
                       0000V  CF              00  FB 00040         CALLS   #0, PUT_ERROR
                                        FD    AB  95 00045 6$:     TSTB    -3(CCB)                         ; 0489
                                              03  19 00048         BLSS    7$
                                        E0    AB  D6 0004A         INCL    -32(CCB)
                          B0   AB        EC    AB  D0 0004D 7$:     MOVL    -20(CCB), -80(CCB)              ; 0496
                               50        D2    AB  3C 00052         MOVZWL  -46(CCB), R0                    ; 0497
                          B4   AB        EC BB40   9E 00056         MOVAB   @-20(CCB)[R0], -76(CCB)
                                              05  0005C         RSB                                     ; 0499

; Routine Size:  93 bytes,    Routine Base:  _FOR$CODE + 0021

;  438          0500  1
```

```
;  440     0501  1 GLOBAL ROUTINE FOR$$REC_RSF0                    ! Read sequential formatted
;  441     0502  1     ! (also FOR$$REC_RSF1, FOR$$REC_RSL0 and FOR$$REC_RSL1)
;  442     0503  1     : JSB_REC0 NOVALUE =
;  443     0504  1
;  444     0505  1 !++
;  445     0506  1 ! FUNCTIONAL DESCRIPTION:
;  446     0507  1 !
;  447     0508  1 !       FOR$$REC_RSF0 (and FOR$$REC_RSF1) reads one formatted sequential record.
;  448     0509  1 !       Increments logical record number after successful read.
;  449     0510  1 !       Then return start and end+1 of user
;  450     0511  1 !       part of record to be processed as input.
;  451     0512  1 !
;  452     0513  1 ! CALLING SEQUENCE:
;  453     0514  1 !
;  454     0515  1 !       JSB FOR$$REC_RSF0 ()
;  455     0516  1 !
;  456     0517  1 ! FORMAL PARAMETERS:
;  457     0518  1 !
;  458     0519  1 !       NONE
;  459     0520  1 !
;  460     0521  1 ! IMPLICIT INPUTS:
;  461     0522  1 !
;  462     0523  1 !       CCB                     Pointer to current logical unit
;  463     0524  1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
;  464     0525  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
;  465     0526  1 !
;  466     0527  1 ! IMPLICIT OUTPUTS:
;  467     0528  1 !
;  468     0529  1 !       LUB$L_LOG_RECNO         Increment logical record number
;  469     0530  1 !                               of next record to be read.
;  470     0531  1 !       LUB$A_BUF_PTR           points to first char of user part of
;  471     0532  1 !                               record buffer.
;  472     0533  1 !       LUB$A_BUF_END           points to end+1 of user part of
;  473     0534  1 !                               record buffer.
;  474     0535  1 !       RAB$W_RSZ               set to read record length, or ZERO
;  475     0536  1 !                               if error.
;  476     0537  1 !
;  477     0538  1 !       RAB$B_RAC               set to sequential access mode
;  478     0539  1 !
;  479     0540  1 ! ROUTINE VALUE:
;  480     0541  1 !
;  481     0542  1 !       NONE
;  482     0543  1 !
;  483     0544  1 ! SIDE EFFECTS:
;  484     0545  1 !
;  485     0546  1 !       Reads next record from file on this logical unit.
;  486     0547  1 !       SIGNAL_STOPs FOR$_ERRDURREA (39='ERROR DURING READ')
;  487     0548  1 !       SIGNAL_STOPs FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
;  488     0549  1 !       SIGNAL_STOPs FOR$_INPRECTOO if record too big
;  489     0550  1 !--
;  490     0551  1
;  491     0552  2     BEGIN
;  492     0553  2
;  493     0554  2     EXTERNAL REGISTER
;  494     0555  2         CCB : REF $FOR$CCB_DECL;
;  495     0556  2
;  496     0557  2     !+
```

```
;  497          0558  2      ! Read record into buffer using RMS and check for erros
;  498          0559  2      ! If end-of-file, SIGNAL_STOP FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
;  499          0560  2      ! If record too big for record buffer, SIGNAL_STOP FOR$_INPRECTOO.
;  500          0561  2      ! If errors, SIGNAL_STO FOR$_ERRDURREA (39='ERROR DURING READ')
;  501          0562  2      !-
;  502          0563  2
;  503          0564  2      CCB [RAB$B_RAC] = RAB$C_SEQ;
;  504          0565  2
;  505          0566  2      IF NOT $GET (RAB = .CCB) THEN GET_ERROR ();
;  506          0567  2
;  507          0568  2      !+
;  508          0569  2      ! Increment logical record number (for backspace)
;  509          0570  2      ! so it is the number of the next record to be read.
;  510          0571  2      !-
;  511          0572  2
;  512          0573  2      IF NOT .CCB [LUB$V_KEYED] THEN CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
;  513          0574  2
;  514          0575  2      !+
;  515          0576  2      ! Check for End-of-file record in sequential organization files.
;  516          0577  2      ! Length = 1 byte and byte is a control Z.
;  517          0578  2      ! SIGNAL_STOP FOR$_ENDDURREA if so.
;  518          0579  2      !-
;  519          0580  2
;  520          0581  2      IF NOT .CCB [LUB$V_NOTSEQORG]
;  521          0582  2      THEN
;  522          0583  2          IF .CCB [RAB$W_RSZ] EQLU 1
;  523          0584  2          THEN
;  524          0585  2              IF .(.CCB [RAB$L_RBF])<0, 8> EQLU FOR$K_CONTROL_Z
;  525          0586  2              THEN
;  526          0587  2                  FOR$$SIGNAL_STO (FOR$K_ENDDURREA);
;  527          0588  2
;  528          0589  2      !+
;  529          0590  2      ! Return start and end+1 address of record just read
;  530          0591  2      !-
;  531          0592  2
;  532          0593  2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
;  533          0594  2      CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
;  534          0595  2      RETURN;
;  535          0596  1      END;                                    ! End of FOR$$REC_RSF0 and FOR$$REC_RSF1


                                                    .EXTRN  SYS$GET

                                 1E   AB  94 00000 FOR$$REC_RSF0::
                                                    CLRB    30(CCB)                                 ; 0564
                                      5B  DD 00003              PUSHL   CCB                         ; 0566
                00000000G     00                    01  FB 00005          CALLS   #1, SYS$GET       ;
                                      05            50  E8 0000C          BLBS    R0, 1$            ;
                     0000V    CF                    00  FB 0000F          CALLS   #0, GET_ERROR     ;
                                 FD   AB  95 00014 1$:      TSTB    -3(CCB)                         ; 0573
                                      03  19 00017          BLSS    2$                            ;
                                 E0   AB  D6 00019          INCL    -32(CCB)                      ;
          15         A1        AB                03  E0 0001C 2$:      BBS     #3, -95(CCB), 3$    ; 0581
                               01         22   AB  B1 00021          CMPW    34(CCB), #1           ; 0583
                                          0F   12 00025          BNEQ    3$                    ;
                               1A         28   BB  91 00027          CMPB    @40(CCB), #26         ; 0585
```

```
                                          09  12 0002B          BNEQ    3$                                      ;
                                          18  DD 0002D          PUSHL   #24                                     ; 0587
                00000000G   00            01  FB 0002F          CALLS   #1, FOR$$SIGNAL_STO                     ;
                      B0    AB       28   AB  D0 00036 3$:      MOVL    40(CCB), -80(CCB)                       ; 0593
                            50       22   AB  3C 0003B          MOVZWL  34(CCB), R0                             ; 0594
                      B4    AB       28 BB40  9E 0003F          MOVAB   @40(CCB)[R0], -76(CCB)                  ;
                                             05 00045          RSB                                             ; 0596

; Routine Size:  70 bytes,    Routine Base:  _FOR$CODE + 007E

;  536          0597  1
```

```
;  538          0598  1 GLOBAL ROUTINE FOR$$REC_RSF9                  ! Read sequential formatted
;  539          0599  1     ! (also RSL9, RMF0, RMF9, WMF0, RD9)
;  540          0600  1     : JSB_REC9 NOVALUE =
;  541          0601  1
;  542          0602  1 !++
;  543          0603  1 ! FUNCTIONAL DESCRIPTION:
;  544          0604  1 !
;  545          0605  1 !       FOR$$REC_RSF9 is a no-op!
;  546          0606  1 !
;  547          0607  1 ! CALLING SEQUENCE:
;  548          0608  1 !
;  549          0609  1 !       JSB FOR$$REC_RSF9 ()
;  550          0610  1 !
;  551          0611  1 ! FORMAL PARAMETERS:
;  552          0612  1 !
;  553          0613  1 !       NONE
;  554          0614  1 !
;  555          0615  1 ! IMPLICIT INPUTS:
;  556          0616  1 !
;  557          0617  1 !       NONE
;  558          0618  1 !
;  559          0619  1 ! IMPLICIT OUTPUTS:
;  560          0620  1 !
;  561          0621  1 ! ROUTINE VALUE:
;  562          0622  1 !
;  563          0623  1 !       NONE
;  564          0624  1 !
;  565          0625  1 ! SIDE EFFECTS:
;  566          0626  1 !
;  567          0627  1 !--
;  568          0628  1
;  569          0629  1     RETURN;


                                        05 00000 FOR$$REC_RSF9::
                                                 RSB                                    ; 0629

; Routine Size:  1 bytes,    Routine Base:  _FOR$CODE + 00C4

;  570          0630  1
```

```
;  572          0631  1 GLOBAL ROUTINE FOR$$REC_WSU0                    ! Write sequential unformatted
;  573          0632  1     : JSB_REC0 NOVALUE =
;  574          0633  1
;  575          0634  1 !++
;  576          0635  1 ! FUNCTIONAL DESCRIPTION:
;  577          0636  1 !
;  578          0637  1 !       Initialize call for write sequential unformatted.  FOR$$REC_WSU0
;  579          0638  1 !       initializes the output buffer and returns start and end+1 of user
;  580          0639  1 !       part of record buffer to be filled by caller.
;  581          0640  1 !       Handles segmented and unsegmented records.
;  582          0641  1 !
;  583          0642  1 ! CALLING SEQUENCE:
;  584          0643  1 !
;  585          0644  1 !       JSB FOR$$REC_WSU0 ()
;  586          0645  1 !
;  587          0646  1 ! FORMAL PARAMETERS:
;  588          0647  1 !
;  589          0648  1 !       NONE
;  590          0649  1 !
;  591          0650  1 ! IMPLICIT INPUTS:
;  592          0651  1 !
;  593          0652  1 !       CCB                     Pointer to current logical unit
;  594          0653  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  595          0654  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
;  596          0655  1 !
;  597          0656  1 ! IMPLICIT OUTPUTS:
;  598          0657  1 !
;  599          0658  1 !       LUB$A_BUF_PTR           Points to start of user part of record buffer
;  600          0659  1 !       LUB$A_BUF_END           Points to end+1 of user part of record buffer
;  601          0660  1 !
;  602          0661  1 !       RAB$B_RAC               Set to correct access mode.
;  603          0662  1 !
;  604          0663  1 ! ROUTINE VALUE:
;  605          0664  1 !
;  606          0665  1 !       NONE
;  607          0666  1 !
;  608          0667  1 ! SIDE EFFECTS:
;  609          0668  1 !
;  610          0669  1 !       NONE
;  611          0670  1 !--
;  612          0671  1
;  613          0672  2     BEGIN
;  614          0673  2
;  615          0674  2     EXTERNAL REGISTER
;  616          0675  2         CCB : REF $FOR$CCB_DECL;
;  617          0676  2
;  618          0677  2     !+
;  619          0678  2     ! Setup first output buffer.  If segmented record control
;  620          0679  2     ! then set first word to 1 as a flag that this is first record
;  621          0680  2     ! in segmented logical record record.  Set first user address
;  622          0681  2     ! to third byte in output buffer.
;  623          0682  2     ! Else (unsegmented) set first user address to first byte in output buffer.
;  624          0683  2     !-
;  625          0684  2
;  626          0685  3     CCB [LUB$A_BUF_PTR] = (IF .CCB [LUB$V_SEGMENTED] THEN
;  627          0686  4         BEGIN
;  628          0687  4         (.CCB [LUB$A_RBUF_ADR])<0, 16> = 1;
```

```
;  629          0688  4          .CCB [LUB$A_RBUF_ADR] + 2
;  630          0689  4          END
;  631          0690  2      ELSE .CCB [LUB$A_RBUF_ADR]);
;  632          0691  2      CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  633          0692  2
;  634          0693  2      !+
;  635          0694  2      ! If user opened with ACCESS='KEYED', set access mode to
;  636          0695  2      ! KEYED, else SEQUENTIAL.
;  637          0696  2      !-
;  638          0697  2
;  639          0698  2      IF .CCB [LUB$V_KEYED] THEN CCB [RAB$B_RAC] = RAB$C_KEY ELSE CCB [RAB$B_RAC] = RAB$C_SEQ;
;  640          0699  2
;  641          0700  2      RETURN;
;  642          0701  1      END;                                    ! End of FOR$$REC_WSU0 routine.

                                    0B    FD   AB          03  E1 00000 FOR$$REC_WSU0::
                                                                        BBC     #3, -3(CCB), 1$                 ; 0685
                                          EC   BB          01  B0 00005         MOVW    #1, @-20(CCB)           ; 0687
                                    50    EC   AB          02  C1 00009         ADDL3   #2, -20(CCB), R0        ; 0688
                                                           04  11 0000E         BRB     2$                      ;
                                               50    EC   AB  D0 00010 1$:      MOVL    -20(CCB), R0            ; 0690
                                          B0   AB          50  D0 00014 2$:      MOVL    R0, -80(CCB)            ; 0685
                                               50    D2   AB  3C 00018         MOVZWL  -46(CCB), R0            ; 0691
                                          B4   AB          EC BB40  9E 0001C         MOVAB   @-20(CCB)[R0], -76(CCB) ;
                                                           FD   AB  95 00022         TSTB    -3(CCB)                 ; 0698
                                                                05  18 00025         BGEQ    3$                      ;
                                          1E   AB          01  90 00027         MOVB    #1, 30(CCB)             ;
                                                           05  0002B         RSB                             ;
                                                      1E   AB  94 0002C 3$:      CLRB    30(CCB)                 ;
                                                           05  0002F         RSB                             ; 0701

; Routine Size:  48 bytes,    Routine Base:  _FOR$CODE + 00C5

;  643          0702  1
```

```
;  645          0703  1 GLOBAL ROUTINE FOR$$REC_WSU1                  ! Write sequential unformatted
;  646          0704  1     : JSB_REC1 NOVALUE =
;  647          0705  1
;  648          0706  1 !++
;  649          0707  1 ! FUNCTIONAL DESCRIPTION:
;  650          0708  1 !
;  651          0709  1 !       Write one sequential unformatted record and initialize for the next
;  652          0710  1 !       FOR$$REC_WSU1 writes one output buffer and then
;  653          0711  1 !       initializes the output buffer and returns start and end+1 of user
;  654          0712  1 !       part of record buffer to be filled by caller.
;  655          0713  1 !       FLR records are null padded.
;  656          0714  1 !       If unsegmented, signal error since only one record allowed per
;  657          0715  1 !       I/O statement.
;  658          0716  1 !       Do not increment logical record number since this is only
;  659          0717  1 !       part of logical record (FOR$$REC_WFU9 will increment).
;  660          0718  1 !
;  661          0719  1 ! CALLING SEQUENCE:
;  662          0720  1 !
;  663          0721  1 !       JSB FOR$$REC_WSU1 ()
;  664          0722  1 !
;  665          0723  1 ! FORMAL PARAMETERS:
;  666          0724  1 !
;  667          0725  1 !       NONE
;  668          0726  1 !
;  669          0727  1 ! IMPLICIT INPUTS:
;  670          0728  1 !
;  671          0729  1 !       CCB                     Pointer to current logical unit
;  672          0730  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  673          0731  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
;  674          0732  1 !       LUB$A_BUF_PTR           Pointer to end+1 of data in uer buffer
;  675          0733  1 !
;  676          0734  1 ! IMPLICIT OUTPUTS:
;  677          0735  1 !
;  678          0736  1 !       LUB$A_BUF_PTR           Points to first char in user part of
;  679          0737  1 !                               record buffer.
;  680          0738  1 !       LUB$A_BUF_END           Points to last+1 char in user part of
;  681          0739  1 !                               record buffer.
;  682          0740  1 !
;  683          0741  1 ! ROUTINE VALUE:
;  684          0742  1 !
;  685          0743  1 !       NONE
;  686          0744  1 !
;  687          0745  1 ! SIDE EFFECTS:
;  688          0746  1 !
;  689          0747  1 !       SIGNAL_STOPs FOR$_OUTSTAOVE if unsegmented record
;  690          0748  1 !--
;  691          0749  1
;  692          0750  2     BEGIN
;  693          0751  2
;  694          0752  2     EXTERNAL REGISTER
;  695          0753  2         CCB : REF $FOR$CCB_DECL;
;  696          0754  2
;  697          0755  2     !+
;  698          0756  2     ! If unsegmented, SIGNAL_STOP FOR$_OUTSTAOVE (66 = 'OUTPUT STATEMENT OVERFLOWED RECORD')
;  699          0757  2     ! since can't go on to next record without segment control.
;  700          0758  2     !-
;  701          0759  2
```

```
;  702     0760  2     IF .CCB [LUB$V_SEGMENTED] EQL 0 THEN FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
;  703     0761  2
;  704     0762  2     !+
;  705     0763  2     ! If fixed length records (FLR), pad with trailing nulls.
;  706     0764  2     ! Set recordsize to actual length of record.
;  707     0765  2     !-
;  708     0766  2
;  709     0767  2     IF .CCB [LUB$V_FIXED]
;  710     0768  2     THEN
;  711     0769  3         BEGIN
;  712     0770  3         CCB [RAB$W_RSZ] = .CCB [LUB$W_RBUF_SIZE];       ! Always set RSZ
;  713     0771  3
;  714     0772  3         IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END] THEN FILL_BUF (0)
;  715     0773  3
;  716     0774  3         END
;  717     0775  2     ELSE
;  718     0776  2         CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_RBUF_ADR];
;  719     0777  2
;  720     0778  2     !+
;  721     0779  2     ! Output buffer to RMS and check for errors
;  722     0780  2     ! If errors, SIGNAL_STO FOR$_ERRDURWRI (38='ERROR DURING WRITE')
;   23     0781  2     !-
;  724     0782  2
;  725     0783  2     CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
;  726     0784  2
;  727     0785  2     IF NOT $PUT (RAB = .CCB) THEN PUT_ERROR ();
;  728     0786  2
;  729     0787  2     !+
;  730     0788  2     ! Setup a subsequent output buffer.  If segmented record control
;  731     0789  2     ! then set first word to 0 as a flag that this is a subsqunt record
;  732     0790  2     ! in segmented logical record (FOR$$REC_WSU9 will set bit 1 to 1 on last record);
;  733     0791  2     ! then set first user address to third byte in output buffer.
;  734     0792  2     ! Else (unsegmented) set first user address to first byte in output buffer.
;  735     0793  2     !-
;  736     0794  2
;  737     0795  3     CCB [LUB$A_BUF_PTR] = (IF .CCB [LUB$V_SEGMENTED] THEN
;  738     0796  4         BEGIN
;  739     0797  4         (.CCB [LUB$A_RBUF_ADR])<0, 16> = 0;
;  740     0798  4         .CCB [LUB$A_RBUF_ADR] + 2
;  741     0799  4         END
;  742     0800  2     ELSE .CCB [LUB$A_RBUF_ADR]);
;  743     0801  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  744     0802  2     RETURN;
;  745     0803  1     END;                                            ! End of FOR$$REC_WSU1


                0B       FD   AB              03   E0 00000 FOR$$REC_WSU1::
                                                           BBS     #3, -3(CCB), 1$                 ; 0760
                         7E        42         8F   9A 00005         MOVZBL  #66, -(SP)              ;
    00000000G    00                          01   FB 00009         CALLS   #1, FOR$$SIGNAL_STO     ;
                15       FD   AB              02   E1 00010 1$:     BBC     #2, -3(CCB), 2$         ; 0767
                         22   AB        D2    AB   B0 00015         MOVW    -46(CCB), 34(CCB)       ; 0770
                         B4   AB        B0    AB   D1 0001A         CMPL    -80(CCB), -76(CCB)      ; 0772
                                              10   1E 0001F         BGEQU   3$                      ;
```

```
                                        7E  D4 00021          CLRL    -(SP)                                                  ;
                              0000V  CF     01  FB 00023          CALLS   #1, FILL_BUF                                           ;
                                     07  11 00028          BRB     3$                                                     ;
                 22  AB         B0  AB  EC  AB  A3 0002A 2$:    SUBW3   -20(CCB), -80(CCB), 34(CCB)                    ; 0776
                               28  AB  EC  AB  D0 00031 3$:    MOVL    -20(CCB), 40(CCB)                              ; 0783
                                     5B  DD 00036          PUSHL   CCB                                                    ; 0785
                        00000000G  00     01  FB 00038          CALLS   #1, SYS$PUT                                            ;
                                   05     50  E8 0003F          BLBS    R0, 4$                                                 ;
                              0000V  CF     00  FB 00042          CALLS   #0, PUT_ERROR                                          ;
                     0A         FD  AB     03  E1 00047 4$:    BBC     #3, -3(CCB), 5$                                ; 0795
                                   EC  BB  B4 0004C          CLRW    @-20(CCB)                                      ; 0797
                     50         EC  AB     02  C1 0004F          ADDL3   #2, -20(CCB), R0                               ; 0798
                                     04  11 00054          BRB     6$                                                     ;
                                50  EC  AB  D0 00056 5$:    MOVL    -20(CCB), R0                                   ; 0800
                               B0  AB     50  D0 0005A 6$:    MOVL    R0, -80(CCB)                                   ; 0795
                                50  D2  AB  3C 0005E          MOVZWL  -46(CCB), R0                                   ; 0801
                               B4  AB  EC BB40  9E 00062          MOVAB   @-20(CCB)[R0], -76(CCB)                                ;
                                     05 00068          RSB                                                            ; 0803

; Routine Size:  105 bytes,    Routine Base:  _FOR$CODE + 00F5

;  746          0804  1
```

```
;  748     0805  1 GLOBAL ROUTINE FOR$$REC_WSU9                    ! Write sequential unformatted
;  749     0806  1     : JSB_REC9 NOVALUE =
;  750     0807  1
;  751     0808  1 !++
;  752     0809  1 ! FUNCTIONAL DESCRIPTION:
;  753     0810  1 !
;  754     0811  1 !       Write last sequential unformatted record.
;  755     0812  1 !       FOR$$REC_WSU9 writes the output buffer
;  756     0813  1 !       If segmented, the first control word is set to 1 as a flag
;  757     0814  1 !       that record is last record of logical segmented record.
;  758     0815  1 !       Otherwise an ordinary write sequential write last record is done.
;  759     0816  1 !       The logical record number is incremented once for entire segmented
;  760     0817  1 !       logical record.
;  761     0818  1 !
;  762     0819  1 ! CALLING SEQUENCE:
;  763     0820  1 !
;  764     0821  1 !       JSB FOR$$REC_WSU9 ()
;  765     0822  1 !
;  766     0823  1 ! FORMAL PARAMETERS:
;  767     0824  1 !
;  768     0825  1 !       NONE
;  769     0826  1 !
;  770     0827  1 ! IMPLICIT INPUTS:
;  771     0828  1 !
;  772     0829  1 !       CCB                     Pointer to current logical unit
;  773     0830  1 !
;  774     0831  1 ! IMPLICIT OUTPUTS:
;  775     0832  1 !
;  776     0833  1 !       LUB$L_LOG_RECNO         Logical record number is incremented
;  777     0834  1 !                               after write is done to be next record.
;  778     0835  1 !
;  779     0836  1 ! ROUTINE VALUE:
;  780     0837  1 !
;  781     0838  1 !       NONE
;  782     0839  1 !
;  783     0840  1 ! SIDE EFFECTS:
;  784     0841  1 !
;  785     0842  1 !       NONE
;  786     0843  1 !--
;  787     0844  1
;  788     0845  2     BEGIN
;  789     0846  2
;  790     0847  2     EXTERNAL REGISTER
;  791     0848  2         CCB : REF $FOR$CCB_DECL;
;  792     0849  2
;  793     0850  2     !+
;  794     0851  2     ! If unsegmented, write this one record as a sequential record.
;  795     0852  2     ! (Fill with trailing nulls if short FLR record).
;  796     0853  2     !-
;  797     0854  2
;  798     0855  2     IF .CCB [LUB$V_SEGMENTED] EQL 0
;  799     0856  2     THEN
;  800     0857  2         JSB_REC9 (FOR$$REC_WSF9)                ! Force JSB linkage on BIND symbol
;  801     0858  2     ELSE
;  802     0859  2
;  803     0860  2     !+
;  804     0861  2     ! Else (segmented control), set bit 1 of first word in buffer as
```

```
;  805          0862  2      ! a mark that this is the last record of segmented record.
;  806          0863  2      ! Note: bit 0 says whether this record is first or subsequent record.
;  807          0864  2      !-
;  808          0865  2
;  809          0866  3          BEGIN
;  810          0867  3          (.CCB [LUB$A_RBUF_ADR])<1, 1> = 1;
;  811          0868  3
;  812          0869  3          !+
;  813          0870  3          ! Write record, check for errors (do not increment logical record number)
;  814          0871  3          !-
;  815          0872  3
;  816          0873  3          FOR$$REC_WSU1 ();
;  817          0874  3
;  818          0875  3          !+
;  819          0876  3          ! Update logical record number (after error check)
;  820          0877  3          ! once for entire logical segmented record.
;  821          0878  3          !-
;  822          0879  3
;  823          0880  3          CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
;  824          0881  2          END;
;  825          0882  2
;  826          0883  2      RETURN;
;  827          0884  1      END;                                    ! End of FOR$$REC_WSU9



                     03       FD   AB            03  E0 00000 FOR$$REC_WSU9::
                                                                BBS     #3, -3(CCB), 1$                 ; 0855
                                                FEBB  31 00005          BRW     FOR$$REC_WSF9           ; 0857
                              EC   BB              02  88 00008 1$:     BISB2   #2, @-20(CCB)           ; 0867
                                                   89  10 0000C         BSBB    FOR$$REC_WSU1           ; 0873
                                                EO   AB  D6 0000E       INCL    -32(CCB)                ; 0880
                                                         05 00011       RSB                             ; 0884

; Routine Size:  18 bytes,    Routine Base:  _FOR$CODE + 015E

;  828          0885  1
```

```
;  830          0886  1 GLOBAL ROUTINE FOR$$REC_RSU0                    ! Read sequential unformatted
;  831          0887  1     ! Called from FOR$$REC_RSU1 if segmented.
;  832          0888  1     : JSB_REC0 NOVALUE =
;  833          0889  1
;  834          0890  1 !++
;  835          0891  1 ! FUNCTIONAL DESCRIPTION:
;  836          0892  1 !
;  837          0893  1 !       FOR$$REC_RSU0 reads one unformatted sequential record.
;  838          0894  1 !       Then return start and end+1 of user
;  839          0895  1 !       part of record to be processed as input.
;  840          0896  1 !       Check for end of file.  If segmented, check for ENDFILE record.
;  841          0897  1 !
;  842          0898  1 ! CALLING SEQUENCE:
;  843          0899  1 !
;  844          0900  1 !       JSB FOR$$REC_RSU0 ()
;  845          0901  1 !
;  846          0902  1 ! FORMAL PARAMETERS:
;  847          0903  1 !
;  848          0904  1 !       NONE
;  849          0905  1 !
;  850          0906  1 ! IMPLICIT INPUTS:
;  851          0907  1 !
;  852          0908  1 !       CCB                     Pointer to current logical unit
;  853          0909  1 !       LUB$A_RBUF_ADR          Address of record buffer set up on OPEN
;  854          0910  1 !       ISB$V_SNGL_ELEM         Flag indicating whether this is a
;  855          0911  1 !                               single-element transfer
;  856          0912  1 !
;  857          0913  1 ! IMPLICIT OUTPUTS:
;  858          0914  1 !
;  859          0915  1 !       ISB$V_LAST_REC          Set to 1 if last record of segmented record read.
;  860          0916  1 !                               Do not increment logical record number
;  861          0917  1 !                               since segmented is one record.
;  862          0918  1 !       LUB$A_BUF_PTR           Pointer to start of user data in buffer
;  863          0919  1 !                               (This routine may bump the pointer over
;  864          0920  1 !                               segmented record control info).
;  865          0921  1 !       RAB$W_RSZ               set to read record length or ZERO
;  866          0922  1 !                               if an error occurs.
;  867          0923  1 !
;  868          0924  1 !       RAB$B_RAC               set to sequential access mode
;  869          0925  1 !
;  870          0926  1 ! ROUTINE VALUE:
;  871          0927  1 !
;  872          0928  1 !       NONE
;  873          0929  1 !
;  874          0930  1 ! SIDE EFFECTS:
;  875          0931  1 !
;  876          0932  1 !       Reads next record from file on this logical unit.
;  877          0933  1 !       SIGNAL_STOPs FOR$_ERRDURREA (39='ERROR DRUING READ')
;  878          0934  1 !       SIGNAL_STOPs FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
;  879          0935  1 !       SIGNAL_STOPs FOR$_INPRECTOO (22) if RMS record too big
;  880          0936  1 !--
;  881          0937  1
;  882          0938  2     BEGIN
;  883          0939  2
;  884          0940  2     EXTERNAL REGISTER
;  885          0941  2         CCB : REF $FOR$CCB_DECL;
;  886          0942  2
```

```
;  887          0943  2      CCB [RAB$B_RAC] = RAB$C_SEQ;
;  888          0944  2
;  889          0945  2      !+
;  890          0946  2      ! Read record into buffer using RMS and check for erros
;  891          0947  2      ! If end-of-file, SIGNAL_STOP FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
;  892          0948  2      ! If record too big for record buffer, SIGNAL_STOP FOR$_INPRECTOO.
;  893          0949  2      ! If errors, SIGNAL_STO FOR$_ERRDURREA (39='ERROR DURING READ')
;  894          0950  2      !
;  895          0951  2      ! RMS$_RTB is not an error unless the record is too big for the
;  896          0952  2      ! record buffer; if it is merely too big for the user element in a
;  897          0953  2      ! single-element transfer, no error nas occurred.
;  898          0954  2      !-
;  899          0955  2
;  900          0956  2      IF NOT $GET (RAB = .CCB) THEN
;  901          0957  3          IF NOT (.CCB [RAB$L_STS] EQL RMS$_RTB
;  902          0958  3                  AND .CCB [RAB$W_RSZ] LEQU .CCB [LUB$W_RBUF_SIZE])
;  903          0959  2              THEN GET_ERROR ();
;  904          0960  2
;  905          0961  2      !+
;  906          0962  2      ! Increment logical record number (for backspace)
;  907          0963  2      ! so it is the number of the next record to be read.
;  908          0964  2      ! Do this only after the record has been successfully read.
;  909          0965  2      !-
;  910          0966  2
;  911          0967  2      IF NOT .CCB [LUB$V_KEYED] THEN CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
;  912          0968  2
;  913          0969  2      !+
;  914          0970  2      ! Return start and end+1 address of record just read
;  915          0971  2      !-
;  916          0972  2
;  917          0973  2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
;  918          0974  2      CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
;  919          0975  2
;  920          0976  2      !+
;  921          0977  2      ! If segmented record, check record format
;  922          0978  2      !-
;  923          0979  2
;  924          0980  2      IF .CCB [LUB$V_SEGMENTED]
;  925          0981  2      THEN
;  926          0982  2
;  927          0983  2      !+
;  928          0984  2      ! If (segmented) record is an ENDFILE record (one byte long and control Z)
;  929          0985  2      ! SIGNAL_STOP (FOR$_ENDDURREA (24='END-FO-FILE DURING READ').
;  930          0986  2      !-
;  931          0987  3
;  932          0988  3          BEGIN
;  933          0989  3
;  934          0990  3          IF .CCB [RAB$W_RSZ] EQL 1
;  935          0991  3          THEN
;  936          0992  3              IF .(.CCB [RAB$L_RBF])<0, 8> EQLU FOR$K_CONTROL_Z
;  937          0993  3              THEN
;  938          0994  3                  FOR$$SIGNAL_STO (FOR$K_ENDDURREA);
;  939          0995  3
;  940          0996  3          !+
;  941          0997  3          ! Check format of segmented record.
;  942          0998  3          ! If record is less than 2 bytes long or bits 15:2 are not zero,
;  943          0999  3          ! SIGNAL_STOPs FOR$_SEGRECFOR (35='SEGMENTED RECORD FORMAT ERROR')
```

```
;  944          1000  3          !-
;  945          1001  3
;  946          1002  3          IF .CCB [RAB$W_RSZ] LSSU 2 OR
;  947          1003  4              (.CCB [RAB$W_RSZ] GEQU 2 AND .(.CCB [RAB$L_RBF])<2, 14> NEQU 0)
;  948          1004  3          THEN
;  949          1005  3              FOR$$SIGNAL_STO (FOR$K_SEGRECFOR);
;  950          1006  3
;  951          1007  3          !+
;  952          1008  3          ! Set buffer pointer beyond end of control word.
;  953          1009  3          ! Check for last record control word (bit 1=1).
;  954          1010  3          ! If last record, set last record of segmented record flag (ISB$V_LAST_REC)
;  955          1011  3          ! and leave logical record number incremented.
;  956          1012  3          ! If not last record, decrement logical record number so not advanced
;  957          1013  3          ! since a segmented logical record is a single record (for backspace).
;  958          1014  3          !-
;  959          1015  3
;  960          1016  3          CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 2;
;  961          1017  3
;  962          1018  3          IF .(.CCB [RAB$L_RBF])<1, 1>
;  963          1019  3          THEN
;  964          1020  3              CCB [ISB$V_LAST_REC] = 1
;  965          1021  3          ELSE
;  966          1022  3              CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] - 1;
;  967          1023  3
;  968          1024  2          END;                                    ! End of segmented processing
;  969          1025  2
;  970          1026  2      RETURN;
;  971          1027  1      END;                                        ! End of FOR$$REC_RSU0

                                    52  DD 00000 FOR$$REC_RSU0::
                                                  PUSHL   R2                              ; 0886
                                 1E   AB  94 00002          CLRB    30(CCB)                 ; 0943
                                 5B   DD 00005              PUSHL   CCB                     ; 0956
                00000000G        00         01  FB 00007          CALLS   #1, SYS$GET     ;
                                 16         50  E8 0000E          BLBS    R0, 2$          ;
                000181A8         8F    08   AB  D1 00011          CMPL    8(CCB), #98728  ; 0957
                                           07  12 00019          BNEQ    1$              ;
                      D2         AB    22   AB  B1 0001B          CMPW    34(CCB), -46(CCB) ; 0958
                                           05  1B 00020          BLEQU   2$              ;
                    0000V        CF         00  FB 00022 1$:      CALLS   #0, GET_ERROR   ; 0959
                                       FD  AB  95 00027 2$:      TSTB    -3(CCB)         ; 0967
                                           03  19 0002A          BLSS    3$              ;
                                       E0  AB  D6 0002C          INCL    -32(CCB)        ;
                                 52        28  AB  9E 0002F 3$:      MOVAB   40(CCB), R2     ; 0973
                           B0    AB              62  D0 00033          MOVL    (R2), -80(CCB)  ;
                                 50        22  AB  3C 00037          MOVZWL  34(CCB), R0     ; 0974
          B4    AB                       62      50  C1 0003B          ADDL3   R0, (R2), -76(CCB) ;
                3E         FD    AB              03  E1 00040          BBC     #3, -3(CCB), 8$ ; 0980
                                 01        22  AB  B1 00045          CMPW    34(CCB), #1     ; 0990
                                           0F  12 00049          BNEQ    4$              ;
                                 1A        00  B2  91 0004B          CMPB    @0(R2), #26     ; 0992
                                           09  12 0004F          BNEQ    4$              ;
                                               18  DD 00051          PUSHL   #24             ; 0994
```

```
                    00000000G  00          01  FB 00053          CALLS   #1, FOR$$SIGNAL_STO                ;
                               02     22    AB  B1 0005A 4$:      CMPW    34(CCB), #2                       ; 1002
                                          08  1F 0005E          BLSSU   5$                                ;
                        FFFC   8F     00    B2  B3 00060          BITW    @0(R2), #65532                    ; 1003
                                          09  13 00066          BEQL    6$                                ;
                                          23  DD 00068 5$:       PUSHL   #35                               ; 1005
                    00000000G  00         01  FB 0006A          CALLS   #1, FOR$$SIGNAL_STO               ;
                          B0   AB         02  C0 00071 6$:       ADDL2   #2, -80(CCB)                      ; 1016
                  06      00   B2         01  E1 00075          BBC     #1, @0(R2), 7$                    ; 1018
                          96   AB         20  88 0007A          BISB2   #32, -106(CCB)                    ; 1020
                                          03  11 0007E          BRB     8$                                ;
                                     E0   AB  D7 00080 7$:       DECL    -32(CCB)                          ; 1022
                                          04  BA 00083 8$:       POPR    #^M<R2>                           ; 1027
                                              05 00085          RSB                                       ;

; Routine Size:  134 bytes,    Routine Base:  _FOR$CODE + 0170

;  972          1028  1
```

```
;  974          1029  1 GLOBAL ROUTINE FOR$$REC_RSU1                    ! Read sequential unformatted
;  975          1030  1     : JSB_REC1 NOVALUE =
;  976          1031  1
;  977          1032  1 !++
;  978          1033  1 ! FUNCTIONAL DESCRIPTION:
;  979          1034  1 !
;  980          1035  1 !       FOR$$REC_RSU1 reads one unformatted sequential record.
;  981          1036  1 !       Then return start and end+1 of user
;  982          1037  1 !       part of record to be processed as input.
;  983          1038  1 !       If no segmented record, then error since only one record allowed.
;  984          1039  1 !       per I/O statement.
;  985          1040  1 !       If last segmented record already read it is also an error.
;  986          1041  1 !
;  987          1042  1 ! CALLING SEQUENCE:
;  988          1043  1 !
;  989          1044  1 !       JSB FOR$$REC_RSU1 ()
;  990          1045  1 !
;  991          1046  1 ! FORMAL PARAMETERS:
;  992          1047  1 !
;  993          1048  1 !       NONE
;  994          1049  1 !
;  995          1050  1 ! IMPLICIT INPUTS:
;  996          1051  1 !
;  997          1052  1 !       CCB                     Pointer to current logical unit
;  998          1053  1 !
;  999          1054  1 ! IMPLICIT OUTPUTS:
; 1000          1055  1 !
; 1001          1056  1 !       LUB$L_LOG_RECNO         Increment logical record number
; 1002          1057  1 !                               of next record to be read.
; 1003          1058  1 !
; 1004          1059  1 ! ROUTINE VALUE:
; 1005          1060  1 !
; 1006          1061  1 !       NONE
; 1007          1062  1 !
; 1008          1063  1 ! SIDE EFFECTS:
; 1009          1064  1 !
; 1010          1065  1 !       Reads next record from file on this logical unit.
; 1011          1066  1 !       SIGNAL_STOs FOR$_ERRDURREA (39='ERROR DURING READ')
; 1012          1067  1 !       SIGNAL_STOPs FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
; 1013          1068  1 !       SIGNAL_STOPs FOR$_SEGRECFOR (35='SEGMENTED RECORD FORMAT ERROR').
; 1014          1069  1 !       SIGNAL_STOPs FOR$_INPSTAREQ (67='INPUT STATEMENT REQUIRED TOO MUCH DATA')
; 1015          1070  1 !--
; 1016          1071  1
; 1017          1072  2     BEGIN
; 1018          1073  2
; 1019          1074  2     EXTERNAL REGISTER
; 1020          1075  2         CCB : REF $FOR$CCB_DECL;
; 1021          1076  2
; 1022          1077  2     !+
; 1023          1078  2     ! If no segmented records, SIGNAL_STOP FOR$_INPSTAREQ (67 ='INPUT STATEMENT REQUIRED TOO MUCH DATA')
; 1024          1079  2     ! since cannot on to next record without segmented control.
; 1025          1080  2     ! If segmented record, go read next record unless already last record.
; 1026          1081  2     !-
; 1027          1082  2
; 1028          1083  2     IF .CCB [LUB$V_SEGMENTED] EQL 0 OR .CCB [ISB$V_LAST_REC] THEN FOR$$SIGNAL_STO (FOR$K_INPSTAREQ);
; 1029          1084  2
; 1030          1085  2     FOR$$REC_RSU0 ();
```

```
;  1031          1086  2     RETURNS;
;  1032          1087  1     END;                                    ! End of FOR$$REC_RSU1


                     05        FD   AB            03  E1 00000 FOR$$REC_RSU1::
                                                                    BBC     #3, -3(CCB), 1$                 ;  1083
                     0B        96   AB            05  E1 00005          BBC     #5, -106(CCB), 2$               ;
                               7E        43      8F  9A 0000A 1$:       MOVZBL  #67, -(SP)                      ;
                        00000000G  00             01  FB 0000E          CALLS   #1, FOR$$SIGNAL_STO             ;
                                                FF62  31 00015 2$:       BRW     FOR$$REC_RSU0                   ;  1085

; Routine Size:  24 bytes,    Routine Base:  _FOR$CODE + 01F6

;  1033          1088  1
```

```
;  1035          1089  1 GLOBAL ROUTINE FOR$$REC_RSU9                    . Read sequential unformatted
;  1036          1090  1     : JSB_REC9 NOVALUE =
;  1037          1091  1
;  1038          1092  1 !++
;  1039          1093  1 ! FUNCTIONAL DESCRIPTION:
;  1040          1094  1 !
;  1041          1095  1 !       FOR$$REC_RSU9 performs termination after last
;  1042          1096  1 !       sequential unformmatted record has been read.
;  1043          1097  1 !       If segmented record and not at last record of segmented record,
;  1044          1098  1 !       keep reading until read last record of segmented record.
;  1045          1099  1 !
;  1046          1100  1 ! CALLING SEQUENCE:
;  1047          1101  1 !
;  1048          1102  1 !       JSB FOR$$REC_RSU9 ()
;  1049          1103  1 !
;  1050          1104  1 ! FORMAL PARAMETERS:
;  1051          1105  1 !
;  1052          1106  1 !       NONE
;  1053          1107  1 !
;  1054          1108  1 ! IMPLICIT INPUTS:
;  1055          1109  1 !
;  1056          1110  1 !       CCB                     Pointer to current logical unit
;  1057          1111  1 !       LUB$V_SEGMENTED         1 = segmented record info is present
;  1058          1112  1 !       ISB$V_LAST_REC          Set to 1 when last record of
;  1059          1113  1 !                               segmented record has been read.
;  1060          1114  1 !
;  1061          1115  1 ! IMPLICIT OUTPUTS:
;  1062          1116  1 !
;  1063          1117  1 ! ROUTINE VALUE:
;  1064          1118  1 !
;  1065          1119  1 !       NONE
;  1066          1120  1 !
;  1067          1121  1 ! SIDE EFFECTS:
;  1068          1122  1 !
;  1069          1123  1 !--
;  1070          1124  1
;  1071          1125  2     BEGIN
;  1072          1126  2
;  1073          1127  2     EXTERNAL REGISTER
;  1074          1128  2         CCB : REF $FOR$CCB_DECL;
;  1075          1129  2
;  1076          1130  2     !+
;  1077          1131  2     ! If segmented record and if not already read last record of segmented record
;  1078          1132  2     ! (ISB$V_LAST_REC = 0) then keep reading records until find last record
;  1079          1133  2     ! (and increment logical record number once).
;  1080          1134  2     !-
;  1081          1135  2
;  1082          1136  2     IF .CCB [LUB$V_SEGMENTED]
;  1083          1137  2     THEN
;  1084          1138  2
;  1085          1139  2         UNTIL .CCB [ISB$V_LAST_REC] DO
;  1086          1140  2             FOR$$REC_RSU1 ();
;  1087          1141  2
;  1088          1142  2     RETURN;
;  1089          1143  1     END;                                          ! End of FOR$$REC_RSU9
```

```
                     09        FD    AB            03  E1 00000 FOR$$REC_RSU9::
                                                                 BBC     #3, -3(CCB), 2$                 ; 1136
                     04        96    AB            05  E0 00005 1$:      BBS     #5, -106(CCB), 2$       ; 1139
                                                     DC  10 0000A         BSBB    FOR$$REC_RSU1           ; 1140
                                                     F7  11 0000C         BRB     1$                      ;
                                                         05 0000E 2$:     RSB                             ; 1143

; Routine Size:  15 bytes,    Routine Base:  _FOR$CODE + 020E

; 1090          1144  1
```

```
;  1092          1145  1 GLOBAL ROUTINE FOR$$REC_WDO                    ! Write direct (formatted and unformatted)
;  1093          1146  1     : JSB_REC0 NOVALUE =
;  1094          1147  1
;  1095          1148  1 !++
;  1096          1149  1 ! FUNCTIONAL DESCRIPTION:
;  1097          1150  1 !
;  1098          1151  1 !       Initialize for FORTRAN direct access (RMS sequential by rec. #) I/O.
;  1099          1152  1 !       Both formatted and unformatted.  Return buffer pointers.
;  1100          1153  1 !
;  1101          1154  1 ! CALLING SEQUENCE:
;  1102          1155  1 !
;  1103          1156  1 !       JSB FOR$$REC_WDO ()
;  1104          1157  1 !
;  1105          1158  1 ! FORMAL PARAMETERS:
;  1106          1159  1 !
;  1107          1160  1 !       NONE
;  1108          1161  1 !
;  1109          1162  1 ! IMPLICIT INPUTS:
;  1110          1163  1 !
;  1111          1164  1 !       CCB                     Pointer to current logical unit
;  1112          1165  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
;  1113          1166  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  1114          1167  1 !
;  1115          1168  1 ! IMPLICIT OUTPUTS:
;  1116          1169  1 !
;  1117          1170  1 !       LUB$A_BUF_PTR           Pointer to start of user part of record
;  1118          1171  1 !                               buffer
;  1119          1172  1 !       LUB$A_BUF_END           Pointer to END+1 of user part of record
;  1120          1173  1 !                               buffer
;  1121          1174  1 !
;  1122          1175  1 ! ROUTINE VALUE:
;  1123          1176  1 !
;  1124          1177  1 !       NONE
;  1125          1178  1 !
;  1126          1179  1 ! SIDE EFFECTS:
;  1127          1180  1 !
;  1128          1181  1 !       NONE
;  1129          1182  1 !--
;  1130          1183  1
;  1131          1184  2     BEGIN
;  1132          1185  2
;  1133          1186  2     EXTERNAL REGISTER
;  1134          1187  2         CCB : REF $FOR$CCB_DECL;
;  1135          1188  2
;  1136          1189  2     !+
;  1137          1190  2     ! Return pointers to the record buffer
;  1138          1191  2     !-
;  1139          1192  2
;  1140          1193  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;  1141          1194  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  1142          1195  2     RETURN;
;  1143          1196  1     END;                                          ! end of routine
```

```
                    B0   AB       EC   AB  D0 00000 FOR$$REC_WD0::
                                                MOVL    -20(CCB), -80(CCB)                  ; 1193
                         50       D2   AB  3C 00005          MOVZWL  -46(CCB), R0                ; 1194
                    B4   AB       EC BB40  9E 00009          MOVAB   @-20(CCB)[R0], -76(CCB)
                                           05 0000F          RSB                                 ; 1196

; Routine Size:  16 bytes,    Routine Base:  _FOR$CODE + 021D

; 1144          1197  1
```

```
;  1146          1198  1 GLOBAL ROUTINE FOR$$REC_WD1                    ! Write direct (formatted and unformatted)
;  1147          1199  1     : JSB_REC1 NOVALUE =
;  1148          1200  1
;  1149          1201  1 !++
;  1150          1202  1 ! FUNCTIONAL DESCRIPTION:
;  1151          1203  1 !
;  1152          1204  1 !       Write current record and reset buffer pointers.  We used to
;  1153          1205  1 !       not allow more than one dirert record.
;  1154          1206  1 !
;  1155          1207  1 ! CALLING SEQUENCE:
;  1156          1208  1 !
;  1157          1209  1 !       JSB FOR$$REC_WD1 ()
;  1158          1210  1 !
;  1159          1211  1 ! FORMAL PARAMETERS:
;  1160          1212  1 !
;  1161          1213  1 !       NONE
;  1162          1214  1 !
;  1163          1215  1 ! IMPLICIT INPUTS:
;  1164          1216  1 !
;  1165          1217  1 !       NONE
;  1166          1218  1 !
;  1167          1219  1 ! IMPLICIT OUTPUTS:
;  1168          1220  1 !
;  1169          1221  1 ! ROUTINE VALUE:
;  1170          1222  1 !
;  1171          1223  1 !       NONE
;  1172          1224  1 !
;  1173          1225  1 ! SIDE EFFECTS:
;  1174          1226  1 !
;  1175          1227  1 !--
;  1176          1228  1
;  1177          1229  2     BEGIN
;  1178          1230  2
;  1179          1231  2     EXTERNAL REGISTER
;  1180          1232  2         CCB : REF $FOR$CCB_DECL;
;  1181          1233  2
;  1182          1234  2     !+
;  1183          1235  2     ! Write a record, then reset pointers.
;  1184          1236  2     !-
;  1185          1237  2
;  1186          1238  2     FOR$$REC_WD9 ();                             ! Write the record
;  1187          1239  2     FOR$$REC_WD0 ();                             ! Reset pointers
;  1188          1240  2     RETURN;
;  1189          1241  1     END;


                                        0000V 30 00000 FOR$$REC_WD1::
                                                       BSBW    FOR$$REC_WD9                    ; 1238
                                          EB  11 00003 BRB     FOR$$REC_WD0                    ; 1239

; Routine Size:  5 bytes,    Routine Base:  _FOR$CODE + 022D

;  1190          1242  1
```

```
;  1192          1243  1 GLOBAL ROUTINE FOR$$REC_WD9                    ! Write direct (formatted and unformatted)
;  1193          1244  1     : JSB_REC9 NOVALUE =
;  1194          1245  1
;  1195          1246  1 !++
;  1196          1247  1 ! FUNCTIONAL DESCRIPTION:
;  1197          1248  1 !
;  1198          1249  1 !       Write a FORTRAN direct access record (RMS sequential by rec. #).
;  1199          1250  1 !       Formatted and unformatted.  Pad out buffer with spaces or nulls
;  1200          1251  1 !       depending on whether formatted or unformatted.
;  1201          1252  1 !
;  1202          1253  1 ! CALLING SEQUENCE:
;  1203          1254  1 !
;  1204          1255  1 !       JSB FOR$$REC_WD9 ()
;  1205          1256  1 !
;  1206          1257  1 ! FORMAL PARAMETERS:
;  1207          1258  1 !
;  1208          1259  1 !       NONE
;  1209          1260  1 !
;  1210          1261  1 !       CCB                     Pointer to current logical unit
;  1211          1262  1 !       LUB$V_UNFORMAT          1 if unformatted, 0 if formatted.
;  1212          1263  1 !       LUB$L_LOG_RECNO         Record number to write
;  1213          1264  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  1214          1265  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
;  1215          1266  1 !       LUB$A_BUF_PTR           Pointer to end+1 of user data in record buffer
;  1216          1267  1 !       ISB$V_SNGL_ELEM         Flag indicating that RAB$W_RSZ and
;  1217          1268  1 !                               RAB$L_RBF have been set up at UDF level
;  1218          1269  1 !
;  1219          1270  1 ! IMPLICIT OUTPUTS:
;  1220          1271  1 !
;  1221          1272  1 !       LUB$L_LOG_RECNO         Updated record number
;  1222          1273  1 !       Ass. Var.               Updated associated variable if present
;  1223          1274  1 !       LUB$V_FIND_LAST         Cleared
;  1224          1275  1 !
;  1225          1276  1 ! ROUTINE VALUE:
;  1226          1277  1 !
;  1227          1278  1 !       NONE
;  1228          1279  1 !
;  1229          1280  1 ! SIDE EFFECTS:
;  1230          1281  1 !
;  1231          1282  1 !       SIGNALSTOPs FOR$_ERRDURWRI if RMS write error.
;  1232          1283  1 !--
;  1233          1284  1
;  1234          1285  2     BEGIN
;  1235          1286  2
;  1236          1287  2     EXTERNAL REGISTER
;  1237          1288  2         CCB : REF $FOR$CCB_DECL;
;  1238          1289  2
;  1239          1290  2     !+
;  1240          1291  2     ! If this is a buffered transfer (ISB$V_SNGL_ELEM = 0), then set up
;  1241          1292  2     ! RAB$W_RSZ and RAB$L_RBF to point to the record buffer.  Otherwise
;  1242          1293  2     ! they have already been set up at UDF level to point to an element.
;  1243          1294  2     !-
;  1244          1295  2
;  1245          1296  2     IF NOT .CCB [ISB$V_SNGL_ELEM]
;  1246          1297  2     THEN
;  1247          1298  3         BEGIN
;  1248          1299  3         IF .CCB [LUB$V_FIXED]
```

```
;  1249          1300  3          THEN
;  1250          1301  4              BEGIN
;  1251          1302  4
;  1252          1303  4              !+
;  1253          1304  4              ! Set RAB$W_RSZ to record size.
;  1254          1305  4              !-
;  1255          1306  4
;  1256          1307  4              CCB [RAB$W_RSZ] = .CCB [LUB$W_RBUF_SIZE];
;  1257          1308  4
;  1258          1309  4              !+
;  1259          1310  4              ! Pad buffer with blanks or nulls to bring record length up to
;  1260          1311  4              ! LUB$W_RBUF_SIZE if not already.
;  1261          1312  4              !-
;  1262          1313  4
;  1263          1314  4              IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END]
;  1264          1315  4              THEN
;  1265          1316  4                  FILL_BUF (
;  1266          1317  4
;  1267          1318  4                      IF .CCB [LUB$V_UNFORMAT] THEN 0 ELSE %C' ');
;  1268          1319  4
;  1269          1320  4              END
;  1270          1321  3          ELSE
;  1271          1322  3              CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_RBUF_ADR];
;  1272          1323  3
;  1273          1324  3          !+
;  1274          1325  3          ! Set RAB record pointer to the record.
;  1275          1326  3          !-
;  1276          1327  3
;  1277          1328  3          CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
;  1278          1329  2          END;
;  1279          1330  2
;  1280          1331  2      !+
;  1281          1332  2      ! Clear LUB$V_FIND_LAST.
;  1282          1333  2      !-
;  1283          1334  2
;  1284          1335  2      CCB [LUB$V_FIND_LAST] = 0;
;  1285          1336  2
;  1286          1337  2      !+
;  1287          1338  2      ! Call RMS to perform the record PUT.
;  1288          1339  2      ! SIGNAL any errors.
;  1289          1340  2      !-
;  1290          1341  2
;  1291          1342  2      IF NOT $PUT (RAB = .CCB) THEN PUT_ERROR ();
;  1292          1343  2
;  1293          1344  2      !+
;  1294          1345  2      ! Increment the record number in the LUB and store
;  1295          1346  2      ! into the associated variable.
;  1296          1347  2      !-
;  1297          1348  2
;  1298          1349  2      CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
;  1299          1350  2      FOR$$ASSOC ();
;  1300          1351  2      RETURN;
;  1301          1352  1      END;                                        ! end of routine
```

```
                 2F      97   AB              04  E0 00000 FOR$$REC_WD9::
                                                        BBS     #4, -105(CCB), 5$                ; 1296
                 1E      FD   AB              02  E1 00005         BBC     #2, -3(CCB), 3$          ; 1299
                         22   AB        D2    AB  B0 0000A         MOVW    -46(CCB), 34(CCB)        ; 1307
                         B4   AB        B0    AB  D1 0000F         CMPL    -80(CCB), -76(CCB)       ; 1314
                                              19  1E 00014         BGEQU   4$
                 04      FD   AB              01  E1 00016         BBC     #1, -3(CCB), 1$          ; 1318
                                              7E  D4 0001B         CLRL    -(SP)
                                              02  11 0001D         BRB     2$
                                              20  DD 0001F 1$:     PUSHL   #32
                     0000V   CF               01  FB 00021 2$:     CALLS   #1, FILL_BUF
                                              07  11 00026         BRB     4$                       ; 1299
             22  AB      B0   AB        EC    AB  A3 00028 3$:     SUBW3   -20(CCB), -80(CCB), 34(CCB) ; 1322
                         28   AB        EC    AB  D0 0002F 4$:     MOVL    -20(CCB), 40(CCB)        ; 1328
                         A0   AB              08  8A 00034 5$:     BICB2   #8, -96(CCB)             ; 1335
                                              5B  DD 00038         PUSHL   CCB                      ; 1342
                00000000G     00              01  FB 0003A         CALLS   #1, SYS$PUT
                              05              50  E8 00041         BLBS    R0, 6$
                     0000V   CF               00  FB 00044         CALLS   #0, PUT_ERROR
                                        E0    AB  D6 00049 6$:     INCL    -32(CCB)                 ; 1349
                00000000G     00              00  FB 0004C         CALLS   #0, FOR$$ASSOC           ; 1350
                                              05  00053         RSB                              ; 1352

; Routine Size:  84 bytes,    Routine Base:  _FOR$CODE + 0232

; 1302          1353  1
```

```
;1304     1354  1 GLOBAL ROUTINE FOR$$REC_RDO                    ! Read direct (formatted and unformatted)
;1305     1355  1     : JSB_REC0 NOVALUE =
;1306     1356  1
;1307     1357  1 !++
;1308     1358  1 ! FUNCTIONAL DESCRIPTION:
;1309     1359  1 !
;1310     1360  1 !       Perform a FORTRAN direct access READ (RMS sequential by rec. #).
;1311     1361  1 !       Formatted and unformatted.  Read record and store
;1312     1362  1 !       ASSOCIATED variable if any.
;1313     1363  1 !
;1314     1364  1 ! CALLING SEQUENCE:
;1315     1365  1 !
;1316     1366  1 !       JSB FOR$$REC_RDO ()
;1317     1367  1 !
;1318     1368  1 ! FORMAL PARAMETERS:
;1319     1369  1 !
;1320     1370  1 !       NONE
;1321     1371  1 !
;1322     1372  1 ! IMPLICIT INPUTS:
;1323     1373  1 !
;1324     1374  1 !       CCB                     Pointer to current logical unit
;1325     1375  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
;1326     1376  1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated at OPEN
;1327     1377  1 !
;1328     1378  1 ! IMPLICIT OUTPUTS:
;1329     1379  1 !
;1330     1380  1 !       LUB$A_BUF_PTR           Pointer to user data in record buffer
;1331     1381  1 !       LUB$A_BUF_END           Pointer to end+1 of user data in record buffer
;1332     1382  1 !       LUB$V_FIND_LAST         Cleared.
;1333     1383  1 !       LUB$L_LOG_RECNO         updated record number
;1334     1384  1 !       ass. var.               updated associated variable (if any)
;1335     1385  1 !       RAB$W_RSZ               set to read record length or ZERO
;1336     1386  1 !                               if error occurs.
;1337     1387  1 !
;1338     1388  1 ! ROUTINE VALUE:
;1339     1389  1 !
;1340     1390  1 !       NONE
;1341     1391  1 !
;1342     1392  1 ! SIDE EFFECTS:
;1343     1393  1 !
;1344     1394  1 !       SIGNALSTOPs FOR$_ERRDUREA if RMS read error
;1345     1395  1 !--
;1346     1396  1
;1347     1397  2     BEGIN
;1348     1398  2
;1349     1399  2     EXTERNAL REGISTER
;1350     1400  2         CCB : REF $FOR$CCB_DECL;
;1351     1401  2
;1352     1402  2     !+
;1353     1403  2     ! Clear LUB$V_FIND_LAST.
;1354     1404  2     !-
;1355     1405  2
;1356     1406  2     CCB [LUB$V_FIND_LAST] = 0;
;1357     1407  2
;1358     1408  2     !+
;1359     1409  2     ! Call RMS to do the record GET.
;1360     1410  2     ! SIGNAL_STOP EOF and RNF errors as ATTEMPT TO ACCESS NON-EXISTANT RECORD.
```

```
;  1361          1411  2      !
;  1362          1412  2      ! RMS$_RTB is not an error unless the record is too big for the
;  1363          1413  2      ! record buffer; if it is merely too big for the user element in an
;  1364          1414  2      ! unbuffered transfer, no error has occurred.
;  1365          1415  2      !-
;  1366          1416  2
;  1367          1417  3      IF NOT $GET (RAB = .CCB)
;  1368          1418  2      THEN
;  1369          1419  3          IF NOT (.CCB [RAB$L_STS] EQL RMS$_RTB
;  1370          1420  3                  AND .CCB [RAB$W_RSZ] LEQU .CCB [LUB$W_RBUF_SIZE])
;  1371          1421  2          THEN
;  1372          1422  3          BEGIN
;  1373          1423  3          !+
;  1374          1424  3          ! Can't use GET_ERROR here because EOF means ATTACCNON.
;  1375          1425  3          !-
;  1376          1426  3
;  1377          1427  3          WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
;  1378          1428  4              BEGIN
;  1379          1429  4              $WAIT (RAB = .CCB);
;  1380          1430  5              $GET (RAB = .CCB)
;  1381          1431  3              END;
;  1382          1432  3
;  1383          1433  3          IF NOT .CCB [RAB$L_STS]
;  1384          1434  3          THEN
;  1385          1435  3              FOR$$SIGNAL_STO (
;  1386          1436  3
;  1387          1437  3                  SELECTONEU .CCB [RAB$L_STS] OF
;  1388          1438  3                      SET
;  1389          1439  3
;  1390          1440  3                      [RMS$_EOF, RMS$_RNF] :
;  1391          1441  3                          FOR$K_ATTACCNON;
;  1392          1442  3
;  1393          1443  3                      [RMS$_RLK] :
;  1394          1444  3                          FOR$K_SPERECLOC;
;  1395          1445  3
;  1396          1446  3                      [RMS$_RTB] :
;  1397          1447  3                          FOR$K_INPRECTOO;
;  1398          1448  3
;  1399          1449  3                      [OTHERWISE] :
;  1400          1450  3                          FOR$K_ERRDURREA;
;  1401          1451  3                      TES);
;  1402          1452  3
;  1403          1453  2              END;
;  1404          1454  2
;  1405          1455  2      !+
;  1406          1456  2      ! Return pointers to the record just read.
;  1407          1457  2      !-
;  1408          1458  2
;  1409          1459  2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
;  1410          1460  2      CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
;  1411          1461  2
;  1412          1462  2      !+
;  1413          1463  2      ! Increment the record number in the LUB and
;  1414          1464  2      ! store it in the associated variable.
;  1415          1465  2      !-
;  1416          1466  2
;  1417          1467  2      CCB [LUB$L_LOG_RECNO] = .CCB [LUB$L_LOG_RECNO] + 1;
```

```
;  1418          1468  2     FOR$$ASSOC ();
;  1419          1469  2     RETURN;
;  1420          1470  1     END;                                    ! end of routine


                                                        .EXTRN  SYS$WAIT

                     A0   AB          08  8A 00000 FOR$$REC_RDO::
                                                    BICB2   #8, -96(CCB)            ; 1406
                                      5B  DD 00004          PUSHL   CCB             ; 1417
                00000000G  00         01  FB 00006          CALLS   #1, SYS$GET
                           6F         50  E8 0000D          BLBS    R0, 8$
                000181A8   8F   08    AB  D1 00010          CMPL    8(CCB), #98728  ; 1419
                                      07  12 00018          BNEQ    1$
                     D2    AB   22    AB  B1 0001A          CMPW    34(CCB), -46(CCB) ; 1420
                                      5E  1B 0001F          BLEQU   8$
                000182DA   8F   08    AB  D1 00021 1$:      CMPL    8(CCB), #99034  ; 1427
                                      14  12 00029          BNEQ    2$
                                      5B  DD 0002B          PUSHL   CCB             ; 1429
                000C0000G  00         01  FB 0002D          CALLS   #1, SYS$WAIT
                                      5B  DD 00034          PUSHL   CCB             ; 1430
                00000000G  00         01  FB 00036          CALLS   #1, SYS$GET
                                      E2  11 0003D          BRB     1$
                           50   08    AB  D0 0003F 2$:      MOVL    8(CCB), R0      ; 1433
                           39         50  E8 00043          BLBS    R0, 8$
                0001827A   8F         50  D1 00046          CMPL    R0, #98938      ; 1440
                                      09  13 0004D          BEQL    3$
                000182B2   8F         50  D1 0004F          CMPL    R0, #98994
                                      04  12 00056          BNEQ    4$
                                      24  DD 00058 3$:      PUSHL   #36
                                      1C  11 0005A          BRB     7$
                000182AA   8F         50  D1 0005C 4$:      CMPL    R0, #98986      ; 1443
                                      04  12 00063          BNEQ    5$
                                      34  DD 00065          PUSHL   #52
                                      0F  11 00067          BRB     7$
                000181A8   8F         50  D1 00069 5$:      CMPL    R0, #98728      ; 1446
                                      04  12 00070          BNEQ    6$
                                      16  DD 00072          PUSHL   #22
                                      02  11 00074          BRB     7$
                                      27  DD 00076 6$:      PUSHL   #39             ; 1449
                00000000G  00         01  FB 00078 7$:      CALLS   #1, FOR$$SIGNAL_STO ; 1437
                     B0    AB   28    AB  D0 0007F 8$:      MOVL    40(CCB), -80(CCB) ; 1459
                           50   22    AB  3C 00084          MOVZWL  34(CCB), R0     ; 1460
                     B4    AB   28 BB40  9E 00088          MOVAB   @40(CCB)[R0], -76(CCB)
                                EO    AB  D6 0008E          INCL    -32(CCB)        ; 1467
                00000000G  00         00  FB 00091          CALLS   #0, FOR$$ASSOC  ; 1468
                                          05 00098          RSB                     ; 1470

; Routine Size:  153 bytes,    Routine Base:  _FOR$CODE + 0286

;  1421          1471  1
```

```
1 GLOBAL ROUTINE FOR$$REC_WSLO                            ! Write list-directed
1      : JSB_RECO NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       FOR$$REC_WSLO prepares a record for list-directed output.
1 !       Then return start and end+1 of user
1 !       part of record to be processed.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       JSB FOR$$REC_WSLO ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       CCB                     Adr. of current LUB/ISB/RAB
1 !       LUB$W_R_MARGIN          Line width set at OPEN time.
1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF $FOR$CCB_DECL;
2
2     !+
2     ! return start and end+1 pointers to caller
2     !-
2
2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_R_MARGIN];
2     RETURN;
1     END;                                                 ! END OF ROUTINE


B0   AB      EC   AB   D0 00000 FOR$$REC_WSLO::
                                  MOVL    -20(CCB), -80(CCB)            ; 1515
     50      D4   AB   3C 00005   MOVZWL  -44(CCB), R0                  ; 1516
B4   AB      EC BB40   9E 00009   MOVAB   @-20(CCB)[R0], -76(CCB)       ;
                       05 0000F   RSB                                   ; 1518
```

; Routine Size: 16 bytes,   Routine Base: _FOR$CODE + 031F

; 1470   1519 1

```
;  1472     1520  1 GLOBAL ROUTINE FOR$$REC_WSL1                  ! Write list-directed
;  1473     1521  1     ! (Also FOR$$REC_WSL9)
;  1474     1522  1     : JSB_REC1 NOVALUE =
;  1475     1523  1
;  1476     1524  1 !++
;  1477     1525  1 ! FUNCTIONAL DESCRIPTION:
;  1478     1526  1 !
;  1479     1527  1 !       Write one list-directed record and initialize for the next
;  1480     1528  1 !       FOR$$REC_WSL1 writes one output buffer and then
;  1481     1529  1 !       initializes the output buffer and returns start and end+1 of user
;  1482     1530  1 !       part of record buffer to be filled by caller.
;  1483     1531  1 !
;  1484     1532  1 ! CALLING SEQUENCE:
;  1485     1533  1 !
;  1486     1534  1 !       JSB FOR$$REC_WSL1 ()
;  1487     1535  1 !
;  1488     1536  1 ! FORMAL PARAMETERS:
;  1489     1537  1 !
;  1490     1538  1 !       NONE
;  1491     1539  1 !
;  1492     1540  1 ! IMPLICIT INPUTS:
;  1493     1541  1 !
;  1494     1542  1 !       CCB                     Pointer to current logical unit
;  1495     1543  1 !       LUB$W_R_MARGIN          Line width set at OPEN time.
;  1496     1544  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
;  1497     1545  1 !       LUB$W_RBUF_SIZE         Buffer size
;  1498     1546  1 !       LUB$V_FIXED             If fixed length records
;  1499     1547  1 !
;  1500     1548  1 ! IMPLICIT OUTPUTS:
;  1501     1549  1 !
;  1502     1550  1 !       LUB$A_BUF_PTR           Pointer to start of user part of record buffer
;  1503     1551  1 !       LUB$A_BUF_END           Pointer to end+1 of user part of record buffer
;  1504     1552  1 !
;  1505     1553  1 ! ROUTINE VALUE:
;  1506     1554  1 !
;  1507     1555  1 !       NONE
;  1508     1556  1 !
;  1509     1557  1 ! SIDE EFFECTS:
;  1510     1558  1 !
;  1511     1559  1 !       Writes one RMS sequential record.
;  1512     1560  1 !       SIGNAL_STOs FOR$_ERRDURWRI on PUT error.
;  1513     1561  1 !--
;  1514     1562  1
;  1515     1563  2     BEGIN
;  1516     1564  2
;  1517     1565  2     EXTERNAL REGISTER
;  1518     1566  2         CCB : REF $FOR$CCB_DECL;
;  1519     1567  2
;  1520     1568  2     !+
;  1521     1569  2     ! If fixed length records, set LUB$A_BUF_END to the actual buffer end
;  1522     1570  2     ! so that blank padding will be done.
;  1523     1571  2     !-
;  1524     1572  2
;  1525     1573  2     IF .CCB [LUB$V_FIXED]
;  1526     1574  2     THEN
;  1527     1575  2         CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  1528     1576  2
```

```
;  1529          1577  2     !+
;  1530          1578  2     ! Call FOR$$REC_WSF1 to perform the record write.
;  1531          1579  2     !-
;  1532          1580  2
;  1533          1581  2     FOR$$REC_WSF1 ();
;  1534          1582  2
;  1535          1583  2     !+
;  1536          1584  2     ! return record buffer pointers to caller
;  1537          1585  2     !-
;  1538          1586  2
;  1539          1587  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;  1540          1588  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_R_MARGIN];
;  1541          1589  2     RETURN;
;  1542          1590  1     END;                                    ! END OF ROUTINE



                    0A      FD   AB          02  E1 00000 FOR$$REC_WSL1::
                                                      BBC     #2, -3(CCB), 1$                 ; 1573
                            50       D2      AB  3C 00005         MOVZWL  -46(CCB), R0            ; 1575
                        B4  AB       EC BB40     9E 00009         MOVAB   @-20(CCB)[R0], -76(CCB)
                                        FCE0     30 0000F 1$:     BSBW    FOR$$REC_WSF1           ; 1581
                        B0  AB       EC      AB  D0 00012         MOVL    -20(CCB), -80(CCB)      ; 1587
                            50       D4      AB  3C 00017         MOVZWL  -44(CCB), R0            ; 1588
                        B4  AB       EC BB40     9E 0001B         MOVAB   @-20(CCB)[R0], -76(CCB)
                                                 05 00021         RSB                             ; 1590

; Routine Size:  34 bytes,    Routine Base:  _FOR$CODE + 032F

;  1543          1591  1
```

```
;1545     1592  1 GLOBAL ROUTINE FOR$$REC_WSNO                    ! Write NAMELIST
;1546     1593  1     : JSB_REC0 NOVALUE =
;1547     1594  1
;1548     1595  1 !++
;1549     1596  1 ! FUNCTIONAL DESCRIPTION:
;1550     1597  1 !
;1551     1598  1 !       FOR$$REC_WSN0 prepares a record for NAMELIST output.
;1552     1599  1 !       Then return start and end+1 of user
;1553     1600  1 !       part of record to be processed.
;1554     1601  1 !
;1555     1602  1 ! CALLING SEQUENCE:
;1556     1603  1 !
;1557     1604  1 !       JSB FOR$$REC_WSN0 ()
;1558     1605  1 !
;1559     1606  1 ! FORMAL PARAMETERS:
;1560     1607  1 !
;1561     1608  1 !       NONE
;1562     1609  1 !
;1563     1610  1 ! IMPLICIT INPUTS:
;1564     1611  1 !
;1565     1612  1 !       CCB                     Adr. of current LUB/ISB/RAB
;1566     1613  1 !       LUB$W_R_MARGIN          Line width set at OPEN time.
;1567     1614  1 !       LUB$W_RBUF_SIZE         Size of buffer
;1568     1615  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
;1569     1616  1 !       LUB$V_FIXED             If fixed length records
;1570     1617  1 !
;1571     1618  1 ! IMPLICIT OUTPUTS:
;1572     1619  1 !
;1573     1620  1 ! ROUTINE VALUE:
;1574     1621  1 !
;1575     1622  1 !       NONE
;1576     1623  1 !
;1577     1624  1 ! SIDE EFFECTS:
;1578     1625  1 !
;1579     1626  1 !       Signals FOR$_OUTSTAOVE if recordsize is zero
;1580     1627  1 !
;1581     1628  1 !--
;1582     1629  1
;1583     1630  2     BEGIN
;1584     1631  2
;1585     1632  2     EXTERNAL REGISTER
;1586     1633  2         CCB : REF $FOR$CCB_DECL;
;1587     1634  2
;1588     1635  2     !+
;1589     1636  2     ! If recordsize is zero, signal FOR$_OUTSTAOVE.
;1590     1637  2     !-
;1591     1638  2
;1592     1639  2     IF .CCB [LUB$W_RBUF_SIZE] EQL 0
;1593     1640  2     THEN
;1594     1641  2         FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
;1595     1642  2
;1596     1643  2     !+
;1597     1644  2     ! return start and end+1 pointers to caller
;1598     1645  2     !-
;1599     1646  2
;1600     1647  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;1601     1648  2     IF .CCB [LUB$V_FIXED]
```

```
;1602          1649  2     THEN
;1603          1650  2         CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE]
;1604          1651  2     ELSE
;1605          1652  2         CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_R_MARGIN];
;1606          1653  2
;1607          1654  2     !+
;1608          1655  2     ! If this is actually a READ statement and if we are using FORTRAN
;1609          1656  2     ! carriagecontrol, write an extra leading space into tne record.
;1610          1657  2     ! This makes the Namelist inquire ('?') feature display items starting
;1611          1658  2     ! in column 2.
;1612          1659  2     !-
;1613          1660  2
;1614          1661  2     IF NOT .CCB [ISB$B_STTM_TYPE] AND   ! low bit 0 if READ
;1615          1662  2        .CCB [LUB$V_FTN]
;1616          1663  2     THEN
;1617          1664  2         CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
;1618          1665  2
;1619          1666  2     RETURN;
;1620          1667  1     END;                                    ! END OF ROUTINE



                                  D2   AB   B5 00000 FOR$$REC_WSNO::
                                                     TSTW    -46(CCB)                    ;1639
                                  0B   12   00003    BNEQ    1$
                            7E    42   8F   9A 00005    MOVZBL  #66, -(SP)                  ;1641
                 00000000G  00        01   FB 00009    CALLS   #1, FOR$$SIGNAL_STO
                       B0   AB   EC   AB   D0 00010 1$:  MOVL    -20(CCB), -80(CCB)          ;1647
                 06    FD   AB        02   E1 00015    BBC     #2, -3(CCB), 2$             ;1648
                      50        D2   AB   3C 0001A    MOVZWL  -46(CCB), R0                ;1650
                                  04   11   0001E    BRB     3$
                      50        D4   AB   3C 00020 2$:  MOVZWL  -44(CCB), R0                ;1652
                 B4   AB        EC BB40   9E 00024 3$:  MOVAB   @-20(CCB)[R0], -76(CCB)
                      0C      FF71   CB   E8 0002A    BLBS    -143(CCB), 4$               ;1661
                              A0   AB   95 0002F    TSTB    -96(CCB)                    ;1662
                                  07   18   00032    BGEQ    4$
                 B0   BB             20   90 00034    MOVB    #32, @-80(CCB)              ;1664
                            B0   AB   D6 00038    INCL    -80(CCB)
                                       05 0003B 4$:  RSB                                 ;1667

; Routine Size:  60 bytes,    Routine Base:  _FOR$CODE + 0351

;1621          1668  1
```

```
;1623   1669  1 GLOBAL ROUTINE FOR$$REC_WSN1                    ! Write NAMELIST
;1624   1670  1     : JSB_REC1 NOVALUE =
;1625   1671  1
;1626   1672  1 !++
;1627   1673  1 ! FUNCTIONAL DESCRIPTION:
;1628   1674  1 !
;1629   1675  1 !       Write one NAMELIST record and initialize for the next.
;1630   1676  1 !       FOR$$REC_WSN1 writes one output buffer and then
;1631   1677  1 !       initializes the output buffer and returns start and end+1 of user
;1632   1678  1 !       part of record buffer to be filled by caller.
;1633   1679  1 !
;1634   1680  1 ! CALLING SEQUENCE:
;1635   1681  1 !
;1636   1682  1 !       JSB FOR$$REC_WSN1 ()
;1637   1683  1 !
;1638   1684  1 ! FORMAL PARAMETERS:
;1639   1685  1 !
;1640   1686  1 !       NONE
;1641   1687  1 !
;1642   1688  1 ! IMPLICIT INPUTS:
;1643   1689  1 !
;1644   1690  1 !       CCB                     Pointer to current logical unit
;1645   1691  1 !       LUB$A_RBUF_ADR          Address of buffer
;1646   1692  1 !       LUB$W_RBUF_SIZE         Size of buffer
;1647   1693  1 !       LUB$W_R_MARGIN          Variable record line width
;1648   1694  1 !       LUB$V_FIXED             If fixed length records
;1649   1695  1 !
;1650   1696  1 ! IMPLICIT OUTPUTS:
;1651   1697  1 !
;1652   1698  1 !       LUB$A_BUF_PTR           Pointer to start of user part of record buffer
;1653   1699  1 !       LUB$A_BUF_END           Pointer to end+1 of user part of record buffer
;1654   1700  1 !
;1655   1701  1 ! ROUTINE VALUE:
;1656   1702  1 !
;1657   1703  1 !       NONE
;1658   1704  1 !
;1659   1705  1 ! SIDE EFFECTS:
;1660   1706  1 !
;1661   1707  1 !       Writes one RMS sequential record.
;1662   1708  1 !       SIGNAL_STOs FOR$_ERRDURWRI on PUT error.
;1663   1709  1 !--
;1664   1710  1
;1665   1711  2     BEGIN
;1666   1712  2
;1667   1713  2     EXTERNAL REGISTER
;1668   1714  2         CCB : REF $FOR$CCB_DECL;
;1669   1715  2
;1670   1716  2                                                                                ;
;1671   1717  2     !+
;1672   1718  2     ! Call FOR$$REC_WSF1 to perform the record write.
;1673   1719  2     !-                                                                         ;
;1674   1720  2
;1675   1721  2     FOR$$REC_WSF1 ();
;1676   1722  2
;1677   1723  2     !+
;1678   1724  2     ! Call FOR$$REC_WSN0 to set up for next record
;1679   1725  2     !-
```

```
;  1680          1726  2
;  1681          1727  2      FOR$$REC_WSN0 ();
;  1682          1728  2
;  1683          1729  2      RETURN;
;  1684          1730  1      END;                                    .  END OF ROUTINE


                                        FC91   30 00000 FOR$$REC_WSN1::
                                                        BSBW    FOR$$REC_WSF1                   ;  1721
                                          BF   11 00003         BRB     FOR$$REC_WSN0           ;  1727

;  Routine Size:  5 bytes,    Routine Base:  _FOR$CODE + 038D

;  1685          1731  1
```

```
;  1687          1732  1 GLOBAL ROUTINE FOR$$REC_RMF1                  ! DECODE entry point for next record
;  1688          1733  1     ! also called as FOR$$REC_WMF1
;  1689          1734  1     : JSB_REC1 NOVALUE =
;  1690          1735  1
;  1691          1736  1 !++
;  1692          1737  1 ! FUNCTIONAL DESCRIPTION:
;  1693          1738  1 !
;  1694          1739  1 !       Since only one "record" is allowed per ENCODE or DECODE statement,
;  1695          1740  1 !       any request for "get next record" is an error.
;  1696          1741  1 !
;  1697          1742  1 ! CALLING SEQUENCE:
;  1698          1743  1 !
;  1699          1744  1 !       JSB FOR$$REC_RMF1 ()
;  1700          1745  1 !
;  1701          1746  1 ! FORMAL PARAMETERS:
;  1702          1747  1 !
;  1703          1748  1 !       NONE
;  1704          1749  1 !
;  1705          1750  1 ! ROUTINE VALUE:
;  1706          1751  1 !
;  1707          1752  1 !       NONE
;  1708          1753  1 !
;  1709          1754  1 ! SIDE EFFECTS:
;  1710          1755  1 !
;  1711          1756  1 !       SIGNAL_STOPs FOR$_TOOMANREC always.
;  1712          1757  1 !--
;  1713          1758  1
;  1714          1759  1     FOR$$SIGNAL_STO (FOR$K_TOOMANREC);


                                1B  DD 00000 FOR$$REC_RMF1::
                                                PUSHL   #27                             ; 1759
                    00000000G   00          01  FB 00002                CALLS   #1, FOR$$SIGNAL_STO     ;
                                                05 00009                RSB             ;

; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 0392

;  1715          1760  1
```

```
GLOBAL ROUTINE FOR$$REC_WMF9                                 ! ENCODE wrap-up
    : JSB_REC9 NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Termination call for ENCODE.
!       Blank pad the end of the users buffer.
!
! FORMAL PARAMETERS:
! CALLING SEQUENCE:
!
!       JSB FOR$$REC_WMF9 ()
!
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       CCB                     Adr. of current LUB/ISB/RAB
!       LUB$A_BUF_PTR           Pointer to end+1 of user data in buffer
!       LUB$A_BUF_END           Pointer to end+1 of buffer
!
! IMPLICIT OUTPUTS:
!
! ROUTINE VALUE:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN

    EXTERNAL REGISTER
        CCB : REF $FOR$CCB_DECL;

    !+
    ! Fill rest of buffer supplied by user if necessary (with spaces).
    !-

    IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END] THEN FILL_BUF (%C' ');

    RETURN;
    END;
```

```
 B4   AB      B0   AB  D1 00000 FOR$$REC_WMF9::
                                        CMPL    -80(CCB), -76(CCB)          ; 1804
                   07  1E 00005         BGEQU   1$                          ;
                   20  DD 00007         PUSHL   #32                         ;
0000V CF           01  FB 00009         CALLS   #1, FILL_BUF                ;
```

```
                        05 0000E 1$:     RSB                                    ; 1807

; Routine Size:  15 bytes,    Routine Base:  _FOR$CODE + 039C

; 1764          1808  1
```

```
;  1766          1809  1 GLOBAL ROUTINE FOR$$REC_WIFO                  ! Write Internal File Formatted - Initialize
;  1767          1810  1     : JSB_REC0 NOVALUE =                      ! (Also FOR$$REC_RIFO)
;  1768          1811  1
;  1769          1812  1 !++
;  1770          1813  1 ! FUNCTIONAL DESCRIPTION:
;  1771          1814  1 !
;  1772          1815  1 !       Initialize the LUB for an internal file read or write.
;  1773          1816  1 !
;  1774          1817  1 ! CALLING SEQUENCE:
;  1775          1818  1 !
;  1776          1819  1 !       JSB FOR$$REC_WIFO ()
;  1777          1820  1 !
;  1778          1821  1 !
;  1779          1822  1 ! IMPLICIT INPUTS:
;  1780          1823  1 !
;  1781          1824  1 !       CCB                     Adr. of current LUB/ISB/RAB
;  1782          1825  1 !       LUB$A_BUF_PTR           Address of the descriptor of the internal
;  1783          1826  1 !                               file, put there by FOR$$IO_BEG.  The only
;  1784          1827  1 !                               allowed descriptor types are DSC$K_DTYPE_T
;  1785          1828  1 !                               and either DSC$K_CLASS_S or DSC$K_CLASS_A.
;  1786          1829  1 !
;  1787          1830  1 ! IMPLICIT OUTPUTS:
;  1788          1831  1 !
;  1789          1832  1 !       LUB$A_BUF_PTR           Pointer to the first character in the
;  1790          1833  1 !                               internal file.
;  1791          1834  1 !       LUB$A_BUF_END           Pointer to the last+1 character in the
;  1792          1835  1 !                               current internal file record.
;  1793          1836  1 !       LUB$W_RBUF_SIZE         Length of one internal file "record".
;  1794          1837  1 !       ISB$A_INTFILEND         Pointer to the last+1 character of the
;  1795          1838  1 !                               internal file.
;  1796          1839  1 !
;  1797          1840  1 ! ROUTINE VALUE:
;  1798          1841  1 !
;  1799          1842  1 !       NONE
;  1800          1843  1 !
;  1801          1844  1 ! SIDE EFFECTS:
;  1802          1845  1 !
;  1803          1846  1 !       FOR$_INVARGFOR if descriptor is not a character scalar or array.
;  1804          1847  1 !--
;  1805          1848  1
;  1806          1849  2     BEGIN
;  1807          1850  2
;  1808          1851  2     EXTERNAL REGISTER
;  1809          1852  2         CCB : REF $FOR$CCB_DECL;
;  1810          1853  2
;  1811          1854  2     LOCAL
;  1812          1855  2         DSC : REF BLOCK [, BYTE];               ! Internal file descriptor
;  1813          1856  2
;  1814          1857  2     !+
;  1815          1858  2     ! Fetch the descriptor and verify that it is of the proper class
;  1816          1859  2     ! and datatype.
;  1817          1860  2     !-
;  1818          1861  2
;  1819          1862  2     DSC = .CCB [LUB$A_BUF_PTR];
;  1820          1863  2
;  1821          1864  3     IF (.DSC [DSC$B_DTYPE] NEQU DSC$K_DTYPE_T OR        !
;  1822          1865  3         ((.DSC [DSC$B_CLASS] NEQU DSC$K_CLASS_S) AND (.DSC [DSC$B_CLASS] NEQU DSC$K_CLASS_A)))
```

```
;  1823          1866  2     THEN
;  1824          1867  2         FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
;  1825          1868  2
;  1826          1869  2     !+
;  1827          1870  2     ! Set up the addresses in the LUB.
;  1828          1871  2     !-
;  1829          1872  2
;  1830          1873  2     CCB [LUB$A_BUF_PTR] = .DSC [DSC$A_POINTER];
;  1831          1874  2     CCB [LUB$W_RBUF_SIZE] = .DSC [DSC$W_LENGTH];
;  1832          1875  2     CCB [LUB$A_BUF_END] = .DSC [DSC$A_POINTER] + .DSC [DSC$W_LENGTH];
;  1833          1876  2
;  1834          1877  2     !+
;  1835          1878  2     ! Calculate the end of the internal file.  If the descriptor is a
;  1836          1879  2     ! scalar, then it is the same as LUB$A_BUF_END.  Otherwise, it
;  1837          1880  2     ! is the end of the array.
;  1838          1881  2     !-
;  1839          1882  2
;  1840          1883  2     IF .DSC [DSC$B_CLASS] EQL DSC$K_CLASS_S
;  1841          1884  2     THEN
;  1842          1885  2         CCB [ISB$A_INTFILEND] = .CCB [LUB$A_BUF_END]
;  1843          1886  2     ELSE
;  1844          1887  2         CCB [ISB$A_INTFILEND] = .DSC [DSC$A_POINTER] + .DSC [DSC$L_ARSIZE];
;  1845          1888  2
;  1846          1889  2     RETURN;
;  1847          1890  1     END;


                                        52  DD 00000 FOR$$REC_WIF0::
                                                    PUSHL   R2                                  ;  1809
                                52      B0  AB  D0 00002                MOVL    -80(CCB), DSC   ;  1862
                                0E      02  A2  91 00006                CMPB    2(DSC), #14     ;  1864
                                            0C  12 0000A                BNEQ    1$              ;
                                01      03  A2  91 0000C                CMPB    3(DSC), #1      ;  1865
                                            0F  13 00010                BEQL    2$              ;
                                04      03  A2  91 00012                CMPB    3(DSC), #4      ;
                                            09  13 00016                BEQL    2$              ;
                                            30  DD 00018 1$:            PUSHL   #48             ;  1867
                        00000000G   00          01  FB 0001A           CALLS   #1, FOR$$SIGNAL_STO ;
                            B0  AB      04  A2  D0 00021 2$:            MOVL    4(DSC), -80(CCB)    ;  1873
                            D2  AB          62  B0 00026                MOVW    (DSC), -46(CCB)     ;  1874
                                50          62  3C 0002A                MOVZWL  (DSC), R0           ;  1875
                            B4  AB      04 B240 9E 0002D                MOVAB   @4(DSC)[R0], -76(CCB)   ;
                                01      03  A2  91 00033                CMPB    3(DSC), #1          ;  1883
                                            07  12 00037                BNEQ    3$                  ;
                            98  AB      B4  AB  D0 00039                MOVL    -76(CCB), -104(CCB) ;  1385
                                            07  11 0003E                BRB     4$                  ;
                98  AB      04  A2      0C  A2  C1 00040 3$:            ADDL3   12(DSC), 4(DSC), -104(CCB)  ;  1887
                                            04  BA 00047 4$:            POPR    #^M<R2>             ;  1890
                                                05 00049                RSB                         ;

; Routine Size:  74 bytes,    Routine Base:  _FOR$CODE + 03AB

;  1848          1891  1
```

```
;1850     1892  1 GLOBAL ROUTINE FOR$$REC_WIF1                  ! Write Internal File Formatted - Next record
;1851     1893  1     : JSB_REC1 NOVALUE =                       ! (Also FOR$$REC_RIF1)
;1852     1894  1
;1853     1895  1 !++
;1854     1896  1 ! FUNCTIONAL DESCRIPTION:
;1855     1897  1 !
;1856     1898  1 !       Fill remainder of record with blanks if WRITE.
;1857     1899  1 !       Advance the LUB pointers to the next internal file "record".
;1858     1900  1 !
;1859     1901  1 ! CALLING SEQUENCE:
;1860     1902  1 !
;1861     1903  1 !       JSB FOR$$REC_WIF1 ()
;1862     1904  1 !
;1863     1905  1 !
;1864     1906  1 ! IMPLICIT INPUTS:
;1865     1907  1 !
;1866     1908  1 !       CCB                     Adr. of current LUB/ISB/RAB
;1867     1909  1 !       LUB$A_BUF_PTR           Pointer to current character in the
;1868     1910  1 !                               current record.
;1869     1911  1 !       LUB$A_BUF_END           Pointer to the last+1 character in the
;1870     1912  1 !                               current record.
;1871     1913  1 !       LUB$W_RBUF_SIZE         Size of one record.
;1872     1914  1 !       ISB$A_INTFILEND         Pointer to last+1 character of the
;1873     1915  1 !                               internal file.
;1874     1916  1 !
;1875     1917  1 ! IMPLICIT OUTPUTS:
;1876     1918  1 !
;1877     1919  1 !       LUB$A_BUF_PTR           Set to first character in the next record.
;1878     1920  1 !       LUB$A_BUF_END           Set to the last+1 character of the next
;1879     1921  1 !                               record.
;1880     1922  1 !
;1881     1923  1 ! ROUTINE VALUE:
;1882     1924  1 !
;1883     1925  1 !       NONE
;1884     1926  1 !
;1885     1927  1 ! SIDE EFFECTS:
;1886     1928  1 !
;1887     1929  1 !       FOR$_ENDDURREA if the "next" record is past the end of the internal
;1888     1930  1 !       file on a READ.
;1889     1931  1 !       FOR$_TOOMANREC if the "next" record is past the end of the internal
;1890     1932  1 !       file on a WRITE.
;1891     1933  1 !--
;1892     1934  1
;1893     1935  2     BEGIN
;1894     1936  2
;1895     1937  2     EXTERNAL REGISTER
;1896     1938  2         CCB : REF $FOR$CCB_DECL;
;1897     1939  2
;1898     1940  2     !+
;1899     1941  2     ! If WRITE, fill remainder of current record with blanks.
;1900     1942  2     !-
;1901     1943  2
;1902     1944  2     IF .CCB [ISB$B_STTM_TYPE]                   ! True if WRITE
;1903     1945  2     THEN
;1904     1946  2
;1905     1947  2         IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END] THEN FILL_BUF (%C' ');
;1906     1948  2
```

```
;  1907          1949  2      !+
;  1908          1950  2      ! Check for the next record being outside the internal file.
;  1909          1951  2      !-
;  1910          1952  2
;  1911          1953  2      IF .CCB [LUB$A_BUF_END] GEQA .CCB [ISB$A_INTFILEND]
;  1912          1954  2      THEN
;  1913          1955  2
;  1914          1956  2          IF .CCB [ISB$B_STTM_TYPE]                ! True if WRITE
;  1915          1957  2          THEN
;  1916          1958  2              FOR$$SIGNAL_STO (FOR$K_TOOMANREC)
;  1917          1959  2          ELSE
;  1918          1960  2              FOR$$SIGNAL_STO (FOR$K_ENDDURREA);
;  1919          1961  2
;  1920          1962  2      !+
;  1921          1963  2      ! Update pointers.
;  1922          1964  2      !-
;  1923          1965  2
;  1924          1966  2      CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_END];
;  1925          1967  2      CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_END];
;  1926          1968  2      CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_END] + .CCB [LUB$W_RBUF_SIZE];
;  1927          1969  2      RETURN;
;  1928          1970  1      END;


                                        0E      FF71    CB  E9 00000 FOR$$REC_WIF1::
                                                                     BLBC    -143(CCB), 1$                   ; 1944
                                    B4  AB        B0    AB  D1 00005         CMPL    -80(CCB), -76(CCB)      ; 1947
                                                        07  1E 0000A         BGEQU   1$                      ;
                                                        20  DD 0000C         PUSHL   #32                     ;
                                                  0000V CF                  01  FB 0000E         CALLS   #1, FILL_BUF            ;
                                                    98  AB          B4      AB  D1 00013 1$:     CMPL    -76(CCB), -104(CCB)     ; 1953
                                                                    12  1F 00018         BLSSU   4$                      ;
                                                        04      FF71    CB  E9 0001A         BLBC    -143(CCB), 2$           ; 1956
                                                                    1B  DD 0001F         PUSHL   #27                     ; 1958
                                                                    02  11 00021         BRB     3$                      ;
                                                                    18  DD 00023 2$:      PUSHL   #24                     ; 1960
                                                00000000G   00      01  FB 00025 3$:      CALLS   #1, FOR$$SIGNAL_STO     ;
                                                      B0    AB          B4  AB  D0 0002C 4$:     MOVL    -76(CCB), -80(CCB)      ; 1966
                                                      BC    AB          B4  AB  D0 00031         MOVL    -76(CCB), -68(CCB)      ; 1967
                                                            50          D2  AB  3C 00036         MOVZWL  -46(CCB), R0            ; 1968
                                                      B4    AB              50  C0 0003A         ADDL2   R0, -76(CCB)            ;
                                                                            05  0003E         RSB                             ; 1970

; Routine Size:  63 bytes,    Routine Base:  _FOR$CODE + 03F5

;  1929          1971  1
```

```
;1931          1972  1 GLOBAL ROUTINE FOR$$REC_RKF0                    ! Read keyed formatted
;1932          1973  1     : JSB_REC0 NOVALUE =
;1933          1974  1
;1934          1975  1 !++
;1935          1976  1 ! FUNCTIONAL DESCRIPTION:
;1936          1977  1 !
;1937          1978  1 !       FOR$$REC_RKF0 reads the first record on a
;1938          1979  1 !       keyed read.
;1939          1980  1 !       Then return start and end+1 of user
;1940          1981  1 !       part of record to be processed as input.
;1941          1982  1 !
;1942          1983  1 ! CALLING SEQUENCE:
;1943          1984  1 !
;1944          1985  1 !       JSB FOR$$REC_RKF0 ()
;1945          1986  1 !
;1946          1987  1 ! FORMAL PARAMETERS:
;1947          1988  1 !
;1948          1989  1 !       NONE
;1949          1990  1 !
;1950          1991  1 ! IMPLICIT INPUTS:
;1951          1992  1 !
;1952          1993  1 !       CCB                     Pointer to current logical unit
;1953          1994  1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
;1954          1995  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
;1955          1996  1 !
;1956          1997  1 ! IMPLICIT OUTPUTS:
;1957          1998  1 !
;1958          1999  1 !       LUB$A_BUF_PTR           points to first char of user part of
;1959          2000  1 !                               record buffer.
;1960          2001  1 !       LUB$A_BUF_END           points to end+1 of user part of
;1961          2002  1 !                               record buffer.
;1962          2003  1 !       RAB$W_RSZ               set to read record length, or ZERO
;1963          2004  1 !                               if error.
;1964          2005  1 !
;1965          2006  1 !       RAB$B_RAC               set to keyed access mode
;1966          2007  1 !
;1967          2008  1 ! ROUTINE VALUE:
;1968          2009  1 !
;1969          2010  1 !       NONE
;1970          2011  1 !
;1971          2012  1 ! SIDE EFFECTS:
;1972          2013  1 !
;1973          2014  1 !       Reads specified record from file on this logical unit.
;1974          2015  1 !       SIGNAL_STOPs FOR$_ERRDURREA (39='ERROR DURING READ')
;1975          2016  1 !       SIGNAL_STOPs FOR$_INPRECTOO if record too big
;1976          2017  1 !--
;1977          2018  1
;1978          2019  2     BEGIN
;1979          2020  2
;1980          2021  2     EXTERNAL REGISTER
;1981          2022  2         CCB : REF $FOR$CCB_DECL;
;1982          2023  2
;1983          2024  2     !+
;1984          2025  2     ! Read record into buffer using RMS and check for erros
;1985          2026  2     ! If end-of-file, SIGNAL_STOP FOR$_ENDDURREA (24='END-OF-FILE DURING READ')
;1986          2027  2     ! If record too big for record buffer, SIGNAL_STOP FOR$_INPRECTOO.
;1987          2028  2     ! If errors, SIGNAL_STO FOR$_ERRDURREA (39='ERROR DURING READ')
```

```
;  1988          2029  2      !-
;  1989          2030  2
;  1990          2031  2      CCB [RAB$B_RAC] = RAB$C_KEY;
;  1991          2032  2
;  1992          2033  2      IF NOT $GET (RAB = .CCB) THEN GET_ERROR ();
;  1993          2034  2
;  1994          2035  2      !+
;  1995          2036  2      ! Return start and end+1 address of record just read
;  1996          2037  2      !-
;  1997          2038  2
;  1998          2039  2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
;  1999          2040  2      CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
;  2000          2041  2      RETURN;
;  2001          2042  1      END;                                    ! End of FOR$$REC_RKF0.



                       1E   AB            01  90 00000 FOR$$REC_RKF0::
                                                       MOVB    #1, 30(CCB)                     ; 2031
                                          5B  DD 00004         PUSHL   CCB                     ; 2033
                00000000G  00            01  FB 00006         CALLS   #1, SYS$GET             ;
                           05            50  E8 0000D         BLBS    R0, 1$                  ;
                     0000V CF            00  FB 00010         CALLS   #0, GET_ERROR           ;
                       B0  AB       28   AB  D0 00015 1$:     MOVL    40(CCB), -80(CCB)       ; 2039
                           50       22   AB  3C 0001A         MOVZWL  34(CCB), R0             ; 2040
                       B4  AB       28 BB40  9E 0001E         MOVAB   @40(CCB)[R0], -76(CCB)  ;
                                             05 00024         RSB                             ; 2042

; Routine Size:  37 bytes,    Routine Base:  _FOR$CODE + 0434

;  2002          2043  1
```

```
;  2004          2044  1 GLOBAL ROUTINE FOR$$REC_WXF9                  ! Rewrite indexed formatted and unformatted
;  2005          2045  1     ! (also FOR$$REC_WX09)
;  2006          2046  1     : JSB_REC9 NOVALUE =
;  2007          2047  1
;  2008          2048  1 !++
;  2009          2049  1 ! FUNCTIONAL DESCRIPTION:
;  2010          2050  1 !
;  2011          2051  1 !       Rewrite the current record on an indexed file.
;  2012          2052  1 !
;  2013          2053  1 ! CALLING SEQUENCE:
;  2014          2054  1 !
;  2015          2055  1 !       JSB FOR$$REC_WXF9 ()
;  2016          2056  1 !
;  2017          2057  1 ! FORMAL PARAMETERS:
;  2018          2058  1 !
;  2019          2059  1 !       NONE
;  2020          2060  1 !
;  2021          2061  1 ! IMPLICIT INPUTS:
;  2022          2062  1 !
;  2023          2063  1 !       CCB                     Pointer to current logical unit
;  2024          2064  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
;  2025          2065  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
;  2026          2066  1 !       LUB$A_BUF_END           points to last char inserted into buffer
;  2027          2067  1 !                               by UDF level I/O.
;  2028          2068  1 !       ISB$V_SNGL_ELEM         Flag indicating that RAB$W_RSZ and
;  2029          2069  1 !                               RAB$L_RBF have been set up at UDF level
;  2030          2070  1 !
;  2031          2071  1 ! IMPLICIT OUTPUTS:
;  2032          2072  1 !
;  2033          2073  1 !       NONE
;  2034          2074  1 !
;  2035          2075  1 ! ROUTINE VALUE:
;  2036          2076  1 !
;  2037          2077  1 !       NONE
;  2038          2078  1 !
;  2039          2079  1 ! SIDE EFFECTS:
;  2040          2080  1 !
;  2041          2081  1 !       NONE
;  2042          2082  1 !--
;  2043          2083  1
;  2044          2084  2     BEGIN
;  2045          2085  2
;  2046          2086  2     EXTERNAL REGISTER
;  2047          2087  2         CCB : REF $FOR$CCB_DECL;
;  2048          2088  2
;  2049          2089  2     !+
;  2050          2090  2     ! If this is a buffered transfer (ISB$V_SNGL_ELEM = 0), then set up
;  2051          2091  2     ! RAB$W_RSZ and RAB$L_RBF to point to the record buffer.  Otherwise
;  2052          2092  2     ! they have already been set up at UDF level to point to an element.
;  2053          2093  2     !-
;  2054          2094  2
;  2055          2095  2     IF NOT .CCB [ISB$V_SNGL_ELEM]
;  2056          2096  2     THEN
;  2057          2097  3         BEGIN
;  2058          2098  3
;  2059          2099  3         !+
;  2060          2100  3         ! If fixed length records (FLR), pad with trailing spaces or
```

```
;  2061          2101  3          ! nulls, depending on whether formatted or unformatted.  Set
;  2062          2102  3          ! recordsize to actual length of record.
;  2063          2103  3          !-
;  2064          2104  3
;  2065          2105  3          IF .CCB [LUB$V_FIXED]
;  2066          2106  3          THEN
;  2067          2107  4              BEGIN
;  2068          2108  4              CCB [RAB$W_RSZ] = .CCB [LUB$W_RBUF_SIZE];   ! Always set RSZ
;  2069          2109  4
;  2070          2110  4              IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_END]
;  2071          2111  4              THEN
;  2072          2112  4                  FILL_BUF (
;  2073          2113  4
;  2074          2114  4                      IF .CCB [LUB$V_UNFORMAT] THEN 0 ELSE %C' ')
;  2075          2115  4
;  2076          2116  4              END
;  2077          2117  3          ELSE
;  2078          2118  3              CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_RBUF_ADR];
;  2079          2119  3          CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
;  2080          2120  2          END;
;  2081          2121  2
;  2082          2122  2      !+
;  2083          2123  2      ! Output buffer to RMS and check for errors
;  2084          2124  2      ! If errors, signal them
;  2085          2125  2      !-
;  2086          2126  2
;  2087          2127  3      IF NOT $UPDATE (RAB = .CCB)
;  2088          2128  2      THEN
;  2089          2129  3          BEGIN
;  2090          2130  3
;  2091          2131  3          WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
;  2092          2132  4              BEGIN
;  2093          2133  4              $WAIT (RAB = .CCB);
;  2094          2134  4              $UPDATE (RAB = .CCB);
;  2095          2135  3              END;
;  2096          2136  3
;  2097          2137  3          IF NOT .CCB [RAB$L_STS]
;  2098          2138  3          THEN
;  2099          2139  3              FOR$$SIGNAL_STO (
;  2100          2140  3
;  2101          2141  3                  SELECTONEU .CCB [RAB$L_STS] OF
;  2102          2142  3                      SET
;  2103          2143  3
;  2104          2144  3                      [RMS$_RNL, RMS$_CUR] :
;  2105          2145  3                          FOR$K_NO_CURREC;
;  2106          2146  3
;  2107          2147  3                      [RMS$_CHG, RMS$_DUP] :
;  2108          2148  3                          FOR$K_INCKEYCHG;
;  2109          2149  3
;  2110          2150  3                      [OTHERWISE] :
;  2111          2151  3                          FOR$K_REWRITERR;
;  2112          2152  3                      TES);
;  2113          2153  3
;  2114          2154  2          END;
;  2115          2155  2
;  2116          2156  2      RETURN;
;  2117          2157  1      END;                                    ! END OF ROUTINE
```

```
                                                        .EXTRN  SYS$UPDATE

                2F      97      AB              04  E0 00000 FOR$$REC_WXF9::
                                                        BBS     #4, -105(CCB), 5$               ; 2095
                1E      FD      AB              02  E1 00005                                   ; 2105
                        22      AB      D2      AB  B0 0000A    MOVW    -46(CCB), 34(CCB)       ; 2108
                        B4      AB      B0      AB  D1 0000F    CMPL    -80(CCB), -76(CCB)      ; 2110
                                                19  1E 00014    BGEQU   4$
                04      FD      AB              01  E1 00016    BBC     #1, -3(CCB), 1$         ; 2114
                                                7E  D4 0001B    CLRL    -(SP)
                                                02  11 0001D    BRB     2$
                                                20  DD 0001F 1$: PUSHL   #32
                        0000V   CF              01  FB 00021 2$: CALLS   #1, FILL_BUF
                                                07  11 00026    BRB     4$                      ; 2110
22      AB              B0      AB      EC      AB  A3 00028 3$: SUBW3   -20(CCB), -80(CCB), 34(CCB)  ; 2118
                        28      AB      EC      AB  D0 0002F 4$: MOVL    -20(CCB), 40(CCB)       ; 2119
                                                5B  DD 00034 5$: PUSHL   CCB                     ; 2127
                00000000G       00              01  FB 00036    CALLS   #1, SYS$UPDATE
                                5B              50  E8 0003D    BLBS    R0, 13$
                000182DA        8F      08      AB  D1 00040 6$: CMPL    8(CCB), #99034          ; 2131
                                                14  12 00048    BNEQ    7$
                                                5B  DD 0004A    PUSHL   CCB                     ; 2133
                00000000G       00              01  FB 0004C    CALLS   #1, SYS$WAIT
                                                5B  DD 00053    PUSHL   CCB                     ; 2134
                00000000G       00              01  FB 00055    CALLS   #1, SYS$UPDATE
                                                E2  11 0005C    BRB     6$                      ; 2131
                                39      08      AB  E8 0005E 7$: BLBS    8(CCB), 13$             ; 2137
                                50      08      AB  D0 00062    MOVL    8(CCB), R0              ; 2141
                000181A0        8F              50  D1 00066    CMPL    R0, #98720              ; 2144
                                                09  13 0006D    BEQL    8$
                000184B4        8F              50  D1 0006F    CMPL    R0, #99508
                                                04  12 00076    BNEQ    9$
                                                35  DD 00078 8$: PUSHL   #53
                                                18  11 0007A    BRB     12$
                0001849C        8F              50  D1 0007C 9$: CMPL    R0, #99484              ; 2147
                                                09  13 00083    BEQL    10$
                000184EC        8F              50  D1 00085    CMPL    R0, #99564
                                                04  12 0008C    BNEQ    11$
                                                32  DD 0008E 10$: PUSHL   #50
                                                02  11 00090    BRB     12$
                                                36  DD 00092 11$: PUSHL   #54                    ; 2150
                00000000G       00              01  FB 00094 12$: CALLS   #1, FOR$$SIGNAL_STO     ; 2141
                                                    05 0009B 13$: RSB                           ; 2157

; Routine Size:  156 bytes,    Routine Base:  _FOR$CODE + 0459

; 2118          2158  1
```

```
;  2120          2159  1 ROUTINE FILL_BUF (                            ! Fill rest of buffer with arg
;  2121          2160  1         FILL_CHAR)                            ! Fill character [by-value]
;  2122          2161  1     : CALL_CCB NOVALUE =
;  2123          2162  1
;  2124          2163  1 !++
;  2125          2164  1 ! FUNCTIONAL DESCRIPTION:
;  2126          2165  1 !
;  2127          2166  1 !       Fill rest of buffer with argument (space or null).
;  2128          2167  1 !       This CALL entry point is provided for JSB routines
;  2129          2168  1 !       to conditionally CALL which are not using R2:R5.
;  2130          2169  1 !
;  2131          2170  1 ! CALLING SEQUENCE:
;  2132          2171  1 !
;  2133          2172  1 !       CALL FILL_BUF (fill_char.rbu.v)
;  2134          2173  1 !
;  2135          2174  1 ! FORMAL PARAMETERS:
;  2136          2175  1 !
;  2137          2176  1 !       fill_char.rbu.v         Fill character
;  2138          2177  1 !
;  2139          2178  1 ! IMPLICIT INPUTS:
;  2140          2179  1 !
;  2141          2180  1 !       CCB                     Adr. of current LUB/ISB/RAB
;  2142          2181  1 !       LUB$A_BUF_PTR           Pointer to end+1 of user data in buffer
;  2143          2182  1 !       LUB$A_BUF_END           Pointer to end+1 of buffer
;  2144          2183  1 !
;  2145          2184  1 ! IMPLICIT OUTPUTS:
;  2146          2185  1 !
;  2147          2186  1 ! ROUTINE VALUE:
;  2148          2187  1 !
;  2149          2188  1 !       NONE
;  2150          2189  1 !
;  2151          2190  1 ! SIDE EFFECTS:
;  2152          2191  1 !
;  2153          2192  1 !       NONE
;  2154          2193  1 !--
;  2155          2194  1
;  2156          2195  2     BEGIN
;  2157          2196  2
;  2158          2197  2     EXTERNAL REGISTER
;  2159          2198  2         CCB : REF $FOR$CCB_DECL;
;  2160          2199  2
;  2161          2200  2     CH$FILL (.FILL_CHAR, .CCB [LUB$A_BUF_END] - .CCB [LUB$A_BUF_PTR], .CCB [LUB$A_BUF_PTR]);
;  2162          2201  2     RETURN;
;  2163          2202  1     END;                                        ! End of FILL_BUF


                                        003C 00000 FILL_BUF:
                                                   .WORD   Save R2,R3,R4,R5                        ; 2159
                    50      B4    AB    B0   AB  C3 00002              SUBL3   -80(CCB), -76(CCB), R0          ; 2200
          50        04      AC          6E       00  2C 00008          MOVC5   #0, (SP), FILL_CHAR, R0, @-80(CCB)  ;
                                                B0  BB     0000E                                                ;
                                                   04 00010              RET                                     ; 2202

; Routine Size:  17 bytes,    Routine Base:  _FOR$CODE + 04F5
```

```
;  2165          2203  1 ROUTINE PUT_ERROR                             ! Here on error in $PUT
;  2166          2204  1     : CALL_CCB NOVALUE =
;  2167          2205  1
;  2168          2206  1 !++
;  2169          2207  1 ! FUNCTIONAL DESCRIPTION:
;  2170          2208  1 !
;  2171          2209  1 !       Here on $PUT errors, check for Record stream active error (RMS$_RSA)
;  2172          2210  1 !       If this error, WAIT until not active and try $PUT again.
;  2173          2211  1 !       This recovers from AST I/O which can occur out of the middle
;  2174          2212  1 !       of synchronous I/O at non-AST level.
;  2175          2213  1 !       Signal FORTRAN specific errors if any.
;  2176          2214  1 !       If any other type of error, SIGNAL_STOP (FOR$_ERRDURWRI).
;  2177          2215  1 !
;  2178          2216  1 ! CALLING SEQUENCE:
;  2179          2217  1 !
;  2180          2218  1 !       JSB PUT_ERROR ()
;  2181          2219  1 !
;  2182          2220  1 ! FORMAL PARAMETERS:
;  2183          2221  1 !
;  2184          2222  1 !       NONE
;  2185          2223  1 !
;  2186          2224  1 ! IMPLICIT INPUTS:
;  2187          2225  1 !
;  2188          2226  1 !       CCB                     Adr. of current LUB/ISB/RAB
;  2189          2227  1 !
;  2190          2228  1 ! IMPLICIT OUTPUTS:
;  2191          2229  1 !
;  2192          2230  1 ! ROUTINE VALUE:
;  2193          2231  1 !
;  2194          2232  1 !       NONE
;  2195          2233  1 !
;  2196          2234  1 ! SIDE EFFECTS:
;  2197          2235  1 !
;  2198          2236  1 !       $WAITs and then tries $PUT again, until success or any error
;  2199          2237  1 !       except record streanm active.
;  2200          2238  1 !--
;  2201          2239  1
;  2202          2240  2     BEGIN
;  2203          2241  2
;  2204          2242  2     EXTERNAL REGISTER
;  2205          2243  2         CCB : REF $FOR$CCB_DECL;
;  2206          2244  2
;  2207          2245  3     WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
;  2208          2246  3         BEGIN
;  2209          2247  3         $WAIT (RAB = .CCB);
;  2210          2248  4         $PUT (RAB = .CCB)
;  2211          2249  2         END;
;  2212          2250  2
;  2213          2251  2     IF NOT .CCB [RAB$L_STS]
;  2214          2252  2     THEN
;  2215          2253  2         FOR$$SIGNAL_STO (
;  2216          2254  2
;  2217          2255  2             SELECTONEU .CCB [RAB$L_STS] OF
;  2218          2256  2                 SET
;  2219          2257  2
;  2220          2258  2                 [RMS$_CHG, RMS$_DUP] :
;  2221          2259  2                     FOR$K_INCKEYCHG;
```

```
;  2222          2260  2
;  2223          2261  2                    [RMS$_RLK] :
;  2224          2262  2                        FOR$K_SPERECLOC;
;  2225          2263  2
;  2226          2264  2                    [OTHERWISE] :
;  2227          2265  2                        FOR$K_ERRDURWRI;
;  2228          2266  2                    TES);
;  2229          2267  2
;  2230          2268  2        RETURN;
;  2231          2269  1        END;                                    ! End of PUT_ERROR


                                        0000 00000 PUT_ERROR:
                                                   .WORD    Save nothing                                    ; 2203
                 000182DA   8F      08   AB  D1 00002 1$:    CMPL    8(CCB), #99034                          ; 2245
                                         14  12 0000A        BNEQ    2$                                      ;
                                         5B  DD 0000C        PUSHL   CCB                                     ; 2247
                 00000000G  00           01  FB 0000E        CALLS   #1, SYS$WAIT                            ;
                                         5B  DD 00015        PUSHL   CCB                                     ; 2248
                 00000000G  00           01  FB 00017        CALLS   #1, SYS$PUT                             ;
                                         E2  11 0001E        BRB     1$                                      ;
                            30      08   AB  E8 00020 2$:    BLBS    8(CCB), 7$                              ; 2251
                            50      08   AB  D0 00024        MOVL    8(CCB), R0                              ; 2255
                 0001849C   8F           50  D1 00028        CMPL    R0, #99484                              ; 2258
                                         09  13 0002F        BEQL    3$                                      ;
                 000184EC   8F           50  D1 00031        CMPL    R0, #99564                              ;
                                         04  12 00038        BNEQ    4$                                      ;
                                         32  DD 0003A 3$:    PUSHL   #50                                     ;
                                         0F  11 0003C        BRB     6$                                      ;
                 000182AA   8F           50  D1 0003E 4$:    CMPL    R0, #98986                              ; 2261
                                         04  12 00045        BNEQ    5$                                      ;
                                         34  DD 00047        PUSHL   #52                                     ;
                                         02  11 00049        BRB     6$                                      ;
                                         26  DD 0004B 5$:    PUSHL   #38                                     ; 2264
                 00000000G  00           01  FB 0004D 6$:    CALLS   #1, FOR$$SIGNAL_STO                     ; 2255
                                             04 00054 7$:    RET                                             ; 2269

; Routine Size:  85 bytes,    Routine Base:  _FOR$CODE + 0506
```

```
;  2233          2270  1 ROUTINE GET_ERROR                                   ! Here on error on $GET
;  2234          2271  1     : CALL_CCB NOVALUE =
;  2235          2272  1
;  2236          2273  1 !++
;  2237          2274  1 ! FUNCTIONAL DESCRIPTION:
;  2238          2275  1 !
;  2239          2276  1 !       Here on $GET errors, check for Record stream active error (RMS$_RSA)
;  2240          2277  1 !       If this error, WAIT until not active and try $GET again.
;  2241          2278  1 !       This recovers from AST I/O which can occur out of the middle
;  2242          2279  1 !       of synchronous I/O at non-AST level.
;  2243          2280  1 !       If any other type of error, SIGNAL_STOP (FOR$_ERRDURRED, FOR$_ENDDURREA,
;  2244          2281  1 !       or FOR$_INPRECTOO) depending on RMS error codes.
;  2245          2282  1 !
;  2246          2283  1 ! CALLING SEQUENCE:
;  2247          2284  1 !
;  2248          2285  1 !       JSB GET_ERROR ()
;  2249          2286  1 !
;  2250          2287  1 ! FORMAL PARAMETERS:
;  2251          2288  1 !
;  2252          2289  1 !       NONE
;  2253          2290  1 !
;  2254          2291  1 ! IMPLICIT INPUTS:
;  2255          2292  1 !
;  2256          2293  1 !       CCB                     Adr. of current LUB/ISB/RAB
;  2257          2294  1 !
;  2258          2295  1 ! IMPLICIT OUTPUTS:
;  2259          2296  1 !
;  2260          2297  1 ! ROUTINE VALUE:
;  2261          2298  1 !
;  2262          2299  1 !       NONE
;  2263          2300  1 !
;  2264          2301  1 ! SIDE EFFECTS:
;  2265          2302  1 !
;  2266          2303  1 !       $WAITs and then tries $GET again, until success or any error
;  2267          2304  1 !       except record streanm active.
;  2268          2305  1 !--
;  2269          2306  1
;  2270          2307  2     BEGIN
;  2271          2308  2
;  2272          2309  2     EXTERNAL REGISTER
;  2273          2310  2         CCB : REF $FOR$CCB_DECL;
;  2274          2311  2
;  2275          2312  2     WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
;  2276          2313  3         BEGIN
;  2277          2314  3         $WAIT (RAB = .CCB);
;  2278          2315  4         $GET (RAB = .CCB)
;  2279          2316  2         END;
;  2280          2317  2
;  2281          2318  2     IF NOT .CCB [RAB$L_STS]
;  2282          2319  2     THEN
;  2283          2320  2         FOR$$SIGNAL_STO (
;  2284          2321  2
;  2285          2322  2             SELECTONEU .CCB [RAB$L_STS] OF
;  2286          2323  2                 SET
;  2287          2324  2
;  2288          2325  2                 [RMS$_EOF] :
;  2289          2326  2                     FOR$K_ENDDURREA;
```

```
; 2290      2327  2
; 2291      2328  2                  [RMS$_RTB] :
; 2292      2329  2                      FOR$K_INPRECTOO;
; 2293      2330  2
; 2294      2331  2                  [RMS$_KEY, RMS$_KRF, RMS$_KSZ] :
; 2295      2332  2                      FOR$K_INVKEYSPE;
; 2296      2333  2
; 2297      2334  2                  [RMS$_RNF] :
; 2298      2335  2                      FOR$K_ATTACCNON;
; 2299      2336  2
; 2300      2337  2                  [RMS$_RLK] :
; 2301      2338  2                      FOR$K_SPERECLOC;
; 2302      2339  2
; 2303      2340  2                  [OTHERWISE] :
; 2304      2341  2                      FOR$K_ERRDURREA;
; 2305      2342  2                  TES);
; 2306      2343  2
; 2307      2344  2          RETURN;
; 2308      2345  1          END;                               ! End of GET_ERROR


                                    0000 00000 GET_ERROR:
                                                       .WORD    Save nothing                 ; 2270
        000182DA  8F       08   AB  D1 00002 1$:       CMPL     8(CCB), #99034               ; 2312
                                14  12 0000A           BNEQ     2$
                                5B  DD 0000C           PUSHL    CCB                          ; 2314
        00000000G 00            01  FB 0000E           CALLS    #1, SYS$WAIT                 ;
                                5B  DD 00015           PUSHL    CCB                          ; 2315
        00000000G 00            01  FB 00017           CALLS    #1, SYS$GET                  ;
                                E2  11 0001E           BRB      1$                           ;
                  60       08   AB  E8 00020 2$:       BLBS     8(CCB), 10$                  ; 2318
                  50       08   AB  D0 00024           MOVL     8(CCB), R0                   ; 2322
        0001827A  8F            50  D1 00028           CMPL     R0, #98938                   ; 2325
                                04  12 0002F           BNEQ     3$                           ;
                                18  DD 00031           PUSHL    #24                          ;
                                48  11 00033           BRB      9$                           ;
        000181A8  8F            50  D1 00035 3$:       CMPL     R0, #98728                   ; 2328
                                04  12 0003C           BNEQ     4$                           ;
                                16  DD 0003E           PUSHL    #22                          ;
                                3B  11 00040           BRB      9$                           ;
        00018594  8F            50  D1 00042 4$:       CMPL     R0, #99732                   ; 2331
                                12  13 00049           BEQL     5$                           ;
        0001859C  8F            50  D1 0004B           CMPL     R0, #99740                   ;
                                09  13 00052           BEQL     5$                           ;
        000185A4  8F            50  D1 00054           CMPL     R0, #99748                   ;
                                04  12 0005B           BNEQ     6$                           ;
                                31  DD 0005D 5$:       PUSHL    #49                          ;
                                1C  11 0005F           BRB      9$                           ;
        000182B2  8F            50  D1 00061 6$:       CMPL     R0, #98994                   ; 2334
                                04  12 00068           BNEQ     7$                           ;
                                24  DD 0006A           PUSHL    #36                          ;
                                0F  11 0006C           BRB      9$                           ;
        000182AA  8F            50  D1 0006E 7$:       CMPL     R0, #98986                   ; 2337
                                04  12 00075           BNEQ     8$                           ;
```

```
                                        34  DD 00077          PUSHL   #52                                    ;
                                        02  11 00079          BRB     9$                                     ;
                                        27  DD 0007B 8$:      PUSHL   #39                                    ; 2340
                    00000000G   00      01  FB 0007D 9$:      CALLS   #1, FOR$$SIGNAL_STO                    ; 2322
                                            04 00084 10$:     RET                                            ; 2345

; Routine Size:  133 bytes,    Routine Base:  _FOR$CODE + 055B

; 2309          2346  1 END
; 2310          2347  1
; 2311          2348  0 ELUDOM
```

```
FOR$$REC_RIL9==     FOR$$REC_RSF9
FOR$$REC_RIL1==     FOR$$REC_WIF1
FOR$$REC_RIL0==     FOR$$REC_WIF0
FOR$$REC_WIL9==     FOR$$REC_WMF9
FOR$$REC_WIL1==     FOR$$REC_WIF1
FOR$$REC_WIL0==     FOR$$REC_WIF0
FOR$$REC_RSN1==     FOR$$REC_RSF0
FOR$$REC_RSN0==     FOR$$REC_RSF0
FOR$$REC_WXU9==     FOR$$REC_WXF9
FOR$$REC_WXU1==     FOR$$REC_RMF1
FOR$$REC_WXU0==     FOR$$REC_WSU0
FOR$$REC_WXF1==     FOR$$REC_RMF1
FOR$$REC_WXF0==     FOR$$REC_WSF0
FOR$$REC_RKU9==     FOR$$REC_RSF9
FOR$$REC_RKU1==     FOR$$REC_RSU1
FOR$$REC_RKU0==     FOR$$REC_RKF0
FOR$$REC_RKF9==     FOR$$REC_RSF9
FOR$$REC_RKF1==     FOR$$REC_RSF0
FOR$$REC_RIF9==     FOR$$REC_RSF9
FOR$$REC_RIF1==     FOR$$REC_WIF1
FOR$$REC_RIF0==     FOR$$REC_WIF0
FOR$$REC_WIF9==     FOR$$REC_WMF9
FOR$$REC_WMF1==     FOR$$REC_RMF1
FOR$$REC_WMF0==     FOR$$REC_RSF9
FOR$$REC_RMF9==     FOR$$REC_RSF9
FOR$$REC_RMF0==     FOR$$REC_RSF9
FOR$$REC_RSL9==     FOR$$REC_RSF9
FOR$$REC_RSL1==     FOR$$REC_RSF0
FOR$$REC_RSL0==     FOR$$REC_RSF0
FOR$$REC_WSL9==     FOR$$REC_WSF1
FOR$$REC_RD9==      FOR$$REC_RSF9
FOR$$REC_RD1==      FOR$$REC_RD0
FOR$$REC_RSF1==     FOR$$REC_RSF0
FOR$$REC_WSF9==     FOR$$REC_WSF1
```

```
                                PSECT SUMMARY

      Name                      Bytes                           Attributes

_FOR$CODE                         1504  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         34          0         581        00:01.0
;   _$255$DUA28:[FORRTL.OBJ]FORLIB.L32;1     711        197         27          52        00:00.6
;   _$255$DUA28:[FORRTL.OBJ]RTLLIB.L32;1      36          0          0           8        00:00.1




;                                      COMMAND QUALIFIERS

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

; Size:         1504 code + 0 data bytes
; Run Time:        00:37.7
; Elapsed Time:    01:20.0
; Lines/CPU Min:    3738
; Lexemes/CPU-Min: 19055
; Memory Used:  142 pages
; Compilation Complete
```

FORREADIF LIS
FORREADIO LIS
FORRECPRO LIS
FORREWRSU LIS
FORSIGNAL LIS
FORREADKO LIS
FORREWIND LIS
FORREADSF LIS
FORREWRSO LIS
FORREADSN LIS
FORREADDU LIS
FORREADSU LIS
FORSECNDS LIS
FORREADIL LIS
FORREADKF LIS
FORREADKU LIS
FORREWRSF LIS
FORREADSL LIS
FORREADSO LIS