```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR         TTT         LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR             TTT         LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR             TTT         LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR             TTT         LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR               TTT         LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR               TTT         LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR               TTT         LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR            TTT         LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR            TTT         LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR            TTT         LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR            TTT         LLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSS     RRR         RRR         TTT         LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSS     RRR         RRR         TTT         LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSS     RRR         RRR         TTT         LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  RRRRRRRR    EEEEEEEEEE    CCCCCCCC  PPPPPPPP    RRRRRRRR      000000
BBBBBBBB      AAAAAA      SSSSSSSS  RRRRRRRR    EEEEEEEEEE    CCCCCCCC  PPPPPPPP    RRRRRRRR      000000
BB      BB  AA      AA  SS          RR      RR  EE          CC          PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS          RR      RR  EE          CC          PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS          RR      RR  EE          CC          PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS          RR      RR  EE          CC          PP      PP  RR      RR  00      00
BBBBBBBB    AA      AA    SSSSSS    RRRRRRRR    EEEEEEEE    CC          PPPPPPPP    RRRRRRRR    00      00
BBBBBBBB    AA      AA    SSSSSS    RRRRRRRR    EEEEEEEE    CC          PPPPPPPP    RRRRRRRR    00      00
BB      BB  AAAAAAAAAA          SS  RR  RR      EE          CC          PP          RR  RR      00      00
BB      BB  AAAAAAAAAA          SS  RR  RR      EE          CC          PP          RR  RR      00      00
BB      BB  AA      AA          SS  RR    RR    EE          CC          PP          RR    RR    00      00      ....
BB      BB  AA      AA          SS  RR    RR    EE          CC          PP          RR    RR    00      00      ....
BBBBBBBB    AA      AA  SSSSSSSS    RR      RR  EEEEEEEEEE    CCCCCCCC  PP          RR      RR    000000        ....
BBBBBBBB    AA      AA  SSSSSSSS    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
```

```
0 MODULE BAS$$REC_PROC (                    . Record processing level of abstraction
0                 IDENT = '1-095'           . File: BASRECPRO.B32 Edit:MDL1095
0                 ) =
1 BEGIN
1 !
1 !*********************************************************************
1 !*                                                                   *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
1 !*  ALL RIGHTS RESERVED.                                             *
1 !*                                                                   *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
1 !*  TRANSFERRED.                                                     *
1 !*                                                                   *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
1 !*  CORPORATION.                                                     *
1 !*                                                                   *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.          *
1 !*                                                                   *
1 !*                                                                   *
1 !*********************************************************************
1
1 !++
1 ! FACILITY: BASIC Support Library - not user callable
1 !
1 ! ABSTRACT:
1 !
1 !       This module implements the record processing level of
1 !       abstraction which is the 3rd level and is called only from
1 !       the user data formatter level (2nd level) when the user
1 !       portion of a record buffer is full (WRITE) or empty
1 !       (READ). This module adds any per record formatting (as
1 !       distinguished from per I/O statement or per I/O list element
1 !       formatting) and then calls RMS ($PUT or $GET). RMS errors
1 !       are converted to BASIC errors and are signaled.
1 !
1 ! ENVIRONMENT:  User access mode; AST level or not.
1 !
1 ! AUTHOR:       Donald G. Petersen;     CREATION DATE: 16-Mar-78
1 !
1 ! MODIFIED BY:
1 !
1 ! 0-61  - Add ATTEMPT TO READ NON-EXISTANT RECORD error to seq. reads.
1 !          JMT 02-Jan-78
1 !       Donald G. Petersen, 16-Mar-78 : VERSION 1-01
1 ! 1-01  - original DGP
1 ! 1-02  - Change to JSB linkages.  DGP 14-Nov-78
1 ! 1-004 - Update copyright notice and add device names to REQUIRE
1 !          files.  JBS 29-NOV-78
1 ! 1-005 - Add BAS$RECOUN to support the Basic RECOUNT function. DGP
1 !          03-Dec-78
```

```
0058  1  ! 1-006 - Dot problem in BAS$RECOUNT.  DGP 04-Dec-78
0059  1  ! 1-007 - Add fudge factor to RECOUNT for the line terminator if input is
0060  1  !          from a terminal.  DGP 04-Dec-78
0061  1  ! 1-008 - Change REQUIRE file names from FOR... to OTS...  JBS 07-DEC-78
0062  1  ! 1-009 - Call BAS$$SIGNAL_IO for RMS errors.  JBS 18-DEC-78
0063  1  ! 1-010 - Add new routines for READ.  DGP 19-Dec-78
0064  1  ! 1-011 - Change references to ISB$A_BUF_PTR, BUF_BEG, BUF_END to LUB.
0065  1  !          DGP 05-Jan-78
0066  1  ! 1-012 - Change to CR format for terminal data files.  DGP 11-Jan-79
0067  1  ! 1-013 - Make a few changes for recursive I/O.  DGP 15-Jan-79
0068  1  ! 1-014 - Remove signalling for ^Z.  Moved to UDF level for INPUT LINE hand-
0069  1  !          ling.  DGP 15-Jan-79
0070  1  ! 1-015 - Change stack frame prefix to BSF$.  JBS 08-FEB-1979
0071  1  ! 1-016 - Add BAS$$REC_GSE (Basic GET sequential).  DGP 19-Feb-79
0072  1  ! 1-017 - Add BAS$$REC_PSE (Basic PUT sequential).  DGP 20-Feb-79
0073  1  ! 1-018 - set RAB$L_RBF in BAS$$REC_PSE.  DGP 20-Feb-79
0074  1  ! 1-019 - Set RAB$L_RBF in BAS$GSE.  DGP 21-Feb-79
0075  1  ! 1-020 - Null fill buffer for GET.  DGP 27-Feb-79
0076  1  ! 1-021 - Add REC routines for FIND, DELETE, UPDATE, RESTORE, SCRATCH.  DGP
0077  1  !          27-Feb-79
0078  1  ! 1-022 - Add BASIOERR.REQ for error handling.  DGP 28-Feb-79
0079  1  ! 1-023 - Add BASUNLOCK and BASFREE.  DGP 28-Feb-79
0080  1  ! 1-024 - Set RAB$L_RBF in BAS$$REC_UPD.  DGP 01-Mar-79
0081  1  ! 1-025 - Add REC_PRE, REC_GRE, REC_FRE.  DGP 02-Mar-79
0082  1  ! 1-026 - More work on relative I/O.  DGP 05-Mar-79
0083  1  ! 1-027 - Update pointer for RE D in Basic Major Frame in RMF9.  DGP 06-Mar-79
0084  1  ! 1-028 - Add support for Basic "foreign buffers".  DGP 27-Mar-79
0085  1  ! 1-029 - Point all GETs and PUTs off to GET_ERROR or PUT_ERROR. DGP 02-Apr-79
0086  1  ! 1-030 - Add more routines to support ISAM.  DGP 03-Apr-79
0087  1  ! 1-031 - Fix PUT sequential to support ISAM.  DGP 04-Apr-79
0088  1  ! 1-032 - Put in indexed I/O stuff. 06-Apr-79
0089  1  ! 1-033 - Bug fixes in indexed.  10-Apr-79  DGP
0090  1  ! 1-034 - Implement the WAIT statement, using LUB$L_WAIT_TIME.
0091  1  !          JBS 10-APR-1979
0092  1  ! 1-035 - Implement the ECHO and NOECHO functions, using LUB$V_NOECHO.
0093  1  !          JBS 17-APR-1979
0094  1  ! 1-036 - Add code to handle single-character input from GET
0095  1  !          SEQUENTIAL.  JBS 17-APR-1979
0096  1  ! 1-037 - Implement the CTRLO and RCTRLO functions, using LUB$V_CCO.
0097  1  !          JBS 19-APR-1979
0098  1  ! 1-038 - Implement the Cancel Typeahead function, using LUB$V_PTA.
0099  1  !          JBS 01-MAY-1979
0100  1  ! 1-039 - Add Basic PRINT USING support.  DGP 15-May-79
0101  1  ! 1-040 - Change BIND to GLOBAL BIND ROUTINE in PRINT USING support.
0102  1  !          JBS 16-MAY-1979
0103  1  ! 1-041 - Add BAS$RECOU_INIT.  JBS 04-JUN-1979
0104  1  ! 1-042 - Add REC level for MAT INPUT.  DGP 05-Jun-79
0105  1  ! 1-043 - Clean up a lot and put real code into Matrix Input routines.  DGP
0106  1  !          14-Jun-79
0107  1  ! 1-044 - Make REC_MIN1 look for continuation character.  DGP 20-Jun-79
0108  1  ! 1-045 - Terminal devices use PRN format for output.  DGP 10-Jul-79
0109  1  ! 1-046 - Add BAS$$NUM_INIT, BAS$$NUM2_INIT, BAS$MAT_LINPUT, BAS$MAT_READ,
0110  1  !          BAS$NUM, BAS$NUM.  DGP 13-Jul-79
0111  1  ! 1-047 - Change ISB$L_MAJ_F_PTR to ISB$A_MAJ_F_PTR.  JBS 24-JUL-1979
0112  1  ! 1-048 - Signal if READ with no DATA.  DGP 07-Aug-79
0113  1  ! 1-049 - Debug MAT I/O.  DGP 07-Aug-79
0114  1  ! 1-050 - STOP a few errors that are being SIGNALled.  DGP 05-Sep-79
```

```
;  115          0115  1  ! 1-051 - FREE and UNLOCK are noops if no record locked.  DGP 06-Sep-79
;  116          0116  1  ! 1-052 - Move NUM_INIT and NUM2_INIT to BAS$MAT_IO, and move BAS$$BLNK_LINE
;  117          0117  1  !          from BAS$MAT_IO to here.  DGP  06-Sep-79
;  118          0118  1  ! 1-053 - Load LUB$A_RBUF_ADR for GET and PUT for Locate mode (RMS).  DGP
;  119          0119  1  !          13-Sep-79
;  120          0120  1  ! 1-054 - Clear the prompt buffer in GET_ERROR.  DGP 17-Sep-79
;  121          0121  1  ! 1-055 - Fix BAS$$BLNK_LINE.  DGP 04-Oct-79
;  122          0122  1  ! 1-056 - Add MAT READ.  DGP 11-Oct-79
;  123          0123  1  ! 1-057 - Add a REC9 routine for MAT PRINT.  DGP 12-Oct-79
;  124          0124  1  ! 1-058 - Add BAS$$REC_MLI1.  DGP 12-Oct-79
;  125          0125  1  ! 1-059 - Fix BAS$$REC_WSL1 to leave the cursor alone.  DGP 02-Nov-79
;  126          0126  1  ! 1-060 - Allow BAS$$REC_WSL1 to accept an argument.  DGP 06-Nov-79
;  127          0127  1  ! 1-061 - GET will only null fill the buffer, if necessary, after a GET.  DGP
;  128          0128  1  !          12-Nov-79
;  129          0129  1  ! 1-062 - BAS$$REC_MPR1 needs an LF if no format char.  DGP 13-Nov-79
;  130          0130  1  ! 1-063 - Use LUB$A_UBF to simplify foreign buffer code.  JBS 13-NOV-1979
;  131          0131  1  ! 1-064 - BAS$$REC_MIN1 should not differentiate between terminal & non-
;  132          0132  1  !          terminal devices.  DGP 14-Nov-79
;  133          0133  1  ! 1-065 - GET relative not null filling the buffer properly.  DGP 29-Nov-79
;  134          0134  1  ! 1-066 - Null fill the buffer before restoring foreign buffer pointers for
;  135          0135  1  !          GET.  DGP 18-Dec-79
;  136          0136  1  ! 1-067 - RMS does not return a terminator in the STV field for files.  DGP
;  137          0137  1  !          03-Jan-80
;  138          0138  1  ! 1-068 - REC_WSL9 should only write a record if the output buffer has some-
;  139          0139  1  !          thing in it to write.  DGP 03-Jan-80
;  140          0140  1  ! 1-069 - Addition to 1-068.  Should also write a record if there was no element
;  141          0141  1  !          transmitter.  DGP 04-Jan-80
;  142          0142  1  ! 1-070 - Unconditionally write a CR in WSLI.  DGP 14-Jan-80
;  143          0143  1  ! 1-071 - Restore "foreign buffers" properly and set RECOUNT in GET Indexed
;  144          0144  1  !          and Relative.  DGP 12-Feb-80
;  145          0145  1  ! 1-072 - Adjust the Global RECOUNT to include the length of an escape sequence.
;  146          0146  1  !          DGP 22-Feb-80
;  147          0147  1  ! 1-073 - A previous edit to fix a problem with foreign buffers reintroduced
;  148          0148  1  !          a problem with only null padding the buffer after a successful GET.
;  149          0149  1  !          DGP 26-Feb-80
;  150          0150  1  ! 1-074 - REC_WSL9 should set VFC2 to BAS$K_NULL (no carriage control) if there
;  151          0151  1  !          is a format character. DGP 26-Feb-80
;  152          0152  1  ! 1-075 - Update the cursor position for INPUT if terminated by an escape.
;  153          0153  1  !          DGP 27-Feb-80
;  154          0154  1  ! 1-076 - When calculting CCPOS (current cursor position) following an INPUT
;  155          0155  1  !          statement, take the prompt string into account.
;  156          0156  1  ! 1-077 - REC_RSL1 is not updating the cursor position correctly.  DGP 04-Mar-80
;  157          0157  1  ! 1-078 - REC_WSL9 should set the 'pre' carriage control for the next record
;  158          0158  1  !          to [F if there is no format charcter 'cuz of recursive I/O.  DGP
;  159          0159  1  !          07-Mar-80
;  160          0160  1  ! 1-079 - Rationalize the CCO and PTA bits.  CCO is now copied from LUB to RAB
;  161          0161  1  !           when initializing for output, and PTA when initializing for input.
;  162          0162  1  !           JBS 31-MAR-1980
;  163          0163  1  ! 1-080 - Clear the dirty bit CCB [LUB$V_OUTBUF_DR] in PUT_ERROR so BAS$CLOSE
;  164          0164  1  !          when invoked by the unwind won't get confused and do a PUT.
;  165          0165  1  !          FM 11-SEP-80
;  166          0166  1  ! 1-081 - Tack on the terminator(s) to the buffer when a GET is done on a
;  167          0167  1  !          terminal device file in BAS$$REC_GSE.
;  168          0168  1  ! 1-082- Add/transfer BAS$WAIT to this module, now wait routines are part of
;  169          0169  1  !          the sharable image.  The routines added are BAS$WAIT,
;  170          0170  1  !          BAS$$READ_WAIT.  We had to make WAIT routines part of the sharable
;  171          0171  1  !          image because WAIT was requested to become a GLOBAL, and routines in
```

```
0172  1 !          this module had to read it.
0173  1 ! 1-083- Only if LUB$B_RAT indicates CR format tack on the CRLF. FM 9-feb-81
0174  1 ! 1-084-  Cursor position not updated correctly if INPUT was
0175  1 !          terminated by an escape - code should check if previous
0176  1 !          PRINT was terminated by a semicolon or comma.  PLL 5-7-81
0177  1 ! 1-085- The purge typeahead function is no longer setting the PTA bit in
0178  1 !         the LUB, so BAS$$REC_RSL0, BAS$$REC_MIN0, and BAS$$REC_GSE don't need
0179  1 !         to check it anymore.  PLL 6-Aug-81
0180  1 ! 1-086 - Add support for RFA access and manual record locking.  PLL 1-Jun-82
0181  1 ! 1-087 - BAS$$REC_GIN and BAS$$REC_FIN should check for a decimal key
0182  1 !          when setting the key size in the RAB.  PLL 6-Jul-1982
0183  1 ! 1-088 - Add support for ANSI INPUT.  If not enough data is supplied, the
0184  1 !          entire INPUT must be restarted.  PLL 29-Jul-1982
0185  1 ! 1-089 - Fix CTRL/O rationalization.  Unconditionally copy whatever its
0186  1 !          state is in the LUB into the RAB, and clear it in the LUB.
0187  1 !          MDL and JBS  10-Aug-1982
0188  1 ! 1-090 - Set buffer pointer (in WSL9) from buffer beginning pointer rather
0189  1 !          than from RBUF_ADR.  This fixes the problem of pointing to the
0190  1 !          wrong buffer when the user enters a CTRL/C while a line is being
0191  1 !          written, and his control-c routine writes a line also.
0192  1 !          MDL and PLL  19-Aug-1982
0193  1 ! 1-091 - In BAS$$REC_RSL1, BAS$$SIGNAL_IO the too little data error instead
0194  1 !          of calling BAS$$STOP_IO.  PLL 27-Sep-1982
0195  1 ! 1-092 - In BAS$$REC_RSL0, the contents of the print buffer should be $PUT
0196  1 !          before the $GET is done, if this is a non-terminal device.  This
0197  1 !          is a requested behavior change that will cause input prompts to
0198  1 !          appear in batch log files.  RMS is making a change concurrently
0199  1 !          that will cause the actual input provided to appear in the batch
0200  1 !          log as well, thus making a batch log an exact duplicate of how
0201  1 !          it would appear if run interactively.  MDL 26-Jul-1983
0202  1 ! 1-093 - check for RMS$_CONTROLC completion status; call new CTRL/C signaller
0203  1 !          if this status is returned.  this change is coordinated with rev.
0204  1 !          2-003 of BAS$CTRLC.  MDL 12-Mar-1984
0205  1 ! 1-094 - for the special case of channel 0, edit 1-092 should reach thru the
0206  1 !          buddy ptr and use the output side of channel 0 to write out the
0207  1 !          prompt string.  MDL 22-Mar-1984
0208  1 ! 1-095 - routines that set and reset record options should reset them BEFORE
0209  1 !          calling error routines, so that subsequent I/O on the channel will
0210  1 !          work properly (if the user handles the error).  MDL 23-Mar-1984
0211  1 !--
0212  1
0213  1 !<BLF/PAGE>
```

```
1 !
1 ! SWITCHES:
1 !
1
1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
1
1 !
1 !       LINKAGES
1 !
1
1 REQUIRE 'RTLIN:OTSLNK';                          ! define all linkages
1
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1     BAS$WAIT : NOVALUE,                          ! Writes into module level OWN WAIT
1     BAS$$READ_WAIT,                              ! Reads the module level OWN WAIT
1     BAS$RECOUNT,                                 ! Support Basic RECOUNT function
1     BAS$$RECOU_INIT : NOVALUE,                   ! Initialize RECOUNT
1     BAS$$BLNK_LINE : CALL_CCB NOVALUE,           ! print a blank line
1     ! write sequential list-directed
1     BAS$$REC_WSL0 : JSB_REC0 NOVALUE,            ! initialize output buffer
1     BAS$$REC_WSL1 : JSB_REC_WSL1 NOVALUE,        ! write all but last record
1     BAS$$REC_WSL9 : JSB_REC9 NOVALUE,            ! write last record
1     ! Mat Linput
1     BAS$$REC_MLI1 : JSB_REC1,                    ! always read another record
1     ! Mat Read
1     BAS$$REC_MRE1 : JSB_REC1,                    ! return a failure
1     ! Mat Print
1     BAS$$REC_MPR1 : JSB_REC1 NOVALUE,            ! write one buffer
1     BAS$$REC_MPR9 : JSB_REC9 NOVALUE,            ! terminate Mat Print
1     ! read sequential list-directed
1     BAS$$REC_RSL0 : JSB_REC0 NOVALUE,            ! read first
1     BAS$$REC_RSL1 : JSB_REC1,                    ! read next
1     BAS$$REC_RSL9 : JSB_REC9 NOVALUE,            ! no-op
1     ! MAT INPUT
1     BAS$$REC_MIN0 : JSB_REC0 NOVALUE,            ! initialize MAT INPUT
1     BAS$$REC_MIN1 : JSB_REC1,                    ! record handler
1     BAS$$REC_MIN9 : JSB_REC9 NOVALUE,            ! terminate MAT INPUT
1     ! read memory list-directed
1     BAS$$REC_RMF0 : JSB_REC0 NOVALUE,            ! initialize read memory
1     BAS$$REC_RMF1 : JSB_REC1 NOVALUE,            ! signal insufficient data
1     BAS$$REC_RMF9 : JSB_REC9 NOVALUE;            ! no-op
1
1 GLOBAL BIND
1     ROUTINE
1     ! write formatted
1     BAS$$REC_WF0 = BAS$$REC_WSL0,
1     BAS$$REC_WF1 = BAS$$REC_WSL1,
1     BAS$$REC_WF9 = BAS$$REC_WSL9;
1
1 FORWARD ROUTINE
1     ! record operations
1     BAS$$REC_GSE : JSB_DO_READ NOVALUE,          ! GET sequential
1     BAS$$REC_PSE : JSB_PUT NOVALUE,              ! PUT sequential
```

```
0699  1     BAS$$REC_FSE : JSB_REC2 NOVALUE,          ! FIND sequential
0700  1     BAS$$REC_FRFA: JSB_REC2 NOVALUE,          ! FIND by RFA
0701  1     BAS$$REC_DSE : JSB_REC0 NOVALUE,          ! DELETE sequential
0702  1     BAS$$REC_UPD : JSB_DO_WRITE NOVALUE,      ! UPDATE
0703  1     BAS$$REC_RSE : JSB_REC0 NOVALUE,          ! RESTORE sequential
0704  1     BAS$$REC_SSE : JSB_REC0 NOVALUE,          ! SCRATCH
0705  1     BAS$$REC_PRE : JSB_PUT NOVALUE,           ! PUT relative with count
0706  1     BAS$$REC_GRE : JSB_DO_READ NOVALUE,       ! GET relative
0707  1     BAS$$REC_GRFA: JSB_DO_READ NOVALUE,       ! GET by RFA
0708  1     BAS$$REC_FRE : JSB_REC0 NOVALUE,          ! FREE relative
0709  1     BAS$$REC_UNL : JSB_REC0 NOVALUE,          ! UNLOCK
0710  1     BAS$$REC_FEE : JSB_REC0 NOVALUE,          ! FREE
0711  1     BAS$$REC_GIN : JSB_REC_IND1 NOVALUE,      ! GET indexed
0712  1     BAS$$REC_FIN : JSB_REC_IND1 NOVALUE,      ! FIND indexed
0713  1     BAS$$REC_RIN : JSB_REC_IND NOVALUE,       ! RESTORE indexed
0714  1     PUT_ERROR : CALL_CCB NOVALUE,             ! error in $PUT
0715  1     GET_ERROR : CALL_CCB NOVALUE;             ! error in $GET
0716  1
0717  1 !
0718  1 ! INCLUDE FILES:
0719  1 !
0720  1
0721  1 REQUIRE 'RTLIN:BASIOERR';                     ! I/O error codes
0774  1
0775  1 REQUIRE 'RTLIN:BASFRAME';                     ! Basic frame offsets
0978  1
0979  1 REQUIRE 'RTLML:OTSISB';                       ! I/O statement block (ISB) offsets
1147  1
1148  1 REQUIRE 'RTLML:OTSLUB';                       ! Logical unit block (LUB) offsets
1288  1
1289  1 REQUIRE 'RTLIN:OTSMAC';                       ! Macros
1483  1
1484  1 REQUIRE 'RTLIN:RTLPSECT';                     ! Define DECLARE_PSECTS macro
1579  1
1580  1 REQUIRE 'RTLML:BASPAR';                       ! BASIC inter-module parameters
1602  1
1603  1 LIBRARY 'RTLSTARLE';                          ! STARLET library for macros and symbols
1604  1
1605  1 !
1606  1 !  MACROS:
1607  1 !
1608  1 !       NONE
1609  1 !
1610  1 !  EQUATED SYMBOLS:
1611  1 !
1612  1
1613  1 LITERAL
1614  1     K_MAT_CONT_CHAR = %X'26',                 ! '&' - Mat Input continuation
1615  1                                               ! character
1616  1     K_STOP = 0,                               ! stop after signalling this error
1617  1     K_SIGNAL = 1;                             ! signal and allow restart
1618  1
1619  1 !
1620  1 !  PSECT DECLARATIONS:
1621  1 !
1622  1 DECLARE_PSECTS (BAS);                         ! declare PSECTs for BAS$ facility
1623  1 !
```

```
1 !  OWN STORAGE:
1
1 OWN
1     RECOUNT : INITIAL (0),
1     WAIT    : WORD INITIAL (0);
1
1 !
1 !  EXTERNAL REFERENCES:
1 !
1
1 EXTERNAL ROUTINE
1     BAS$$SIGNAL : NOVALUE,                          ! Signal a BASIC error
1     BAS$$STOP_IO : NOVALUE,                         ! Signal fatal Basic error
1     BAS$$SIGNAL_IO : NOVALUE,                       ! Signal a BASIC I/O error
1     BAS$$SIGNAL_CTRLC : NOVALUE;                    ! Signal CTRL/C
1
1 !
1
1 EXTERNAL LITERAL
1     BAS$K_OUTOF_DAT : UNSIGNED (8),                 ! out of data (READ)
1     BAS$K_ENDFILDEV : UNSIGNED (8),                 ! end of file on device
1     BAS$K_NOTENODAT : UNSIGNED (8),                 ! not enough data
1     BAS$K_TOOLITDAT : UNSIGNED (8),                 ! ANSI for above
1     BAS$K_RECFILTOO : UNSIGNED (8);                 ! record in file too long
1
1 !
1
```

```
;  357          1651  1 GLOBAL ROUTINE BAS$WAIT (                        !Limit input wait time
;  358          1652  1         TIME                                     !Seconds to limit time
;  359          1653  1     ) : NOVALUE =
;  360          1654  1
;  361          1655  1 !++
;  362          1656  1 ! FUNCTIONAL DESCRIPTION:
;  363          1657  1 !
;  364          1658  1 !       Limits the time any input I/O statement ( INPUT, INPUT LINE, LINPUT,
;  365          1659  1 !       MAT 'all above', GET ) to any terminal will wait. If the user does not
;  366          1660  1 !       reply before the indicated number of seconds an error trap ( which the
;  367          1661  1 !       user can intercept ) will be taken.  WAIT is a module level OWN in this
;  368          1662  1 !       module.
;  369          1663  1 !
;  370          1664  1 !
;  371          1665  1 ! FORMAL PARAMETERS:
;  372          1666  1 !
;  373          1667  1 !       TIME.rl.v       Number of seconds to wait, max.
;  374          1668  1 !
;  375          1669  1 ! IMPLICIT INPUTS:
;  376          1670  1 !
;  377          1671  1 !       The module level OWN WAIT
;  378          1672  1 !
;  379          1673  1 ! IMPLICIT OUTPUTS:
;  380          1674  1 !
;  381          1675  1 !       Writes to the module level OWN WAIT the number of seconds given
;  382          1676  1 !
;  383          1677  1 ! ROUTINE VALUE:
;  384          1678  1 !
;  385          1679  1 !       None
;  386          1680  1 !
;  387          1681  1 ! SIDE EFFECTS:
;  388          1682  1 !
;  389          1683  1 !       None
;  390          1684  1 !
;  391          1685  1 !--
;  392          1686  1
;  393          1687  2     BEGIN
;  394          1688  2 !+
;  395          1689  2 ! If the WAIT time is unreasonable then force it to the acceptable range.
;  396          1690  2 ! This is until the correct error message is cooked up for this error.
;  397          1691  2 ! WAIT is a module level OWN.
;  398          1692  2 !-
;  399          1693  2     WAIT = MIN ( ABS(.TIME) , 255 );
;  400          1694  2     RETURN;
;  401          1695  1     END;                                    !End of BAS$WAIT


                                                        .TITLE  BAS$$REC_PROC
                                                        .IDENT  \1-095\

                                                        .PSECT  _BAS$DATA,NOEXE,  PIC,2

                                00000000  00000 RECOUNT:.LONG   0
                                    0000  00004 WAIT.   .WORD   0

                                                        .EXTRN  BAS$$SIGNAL, BAS$$STOP_IO
                                                        .EXTRN  BAS$$SIGNAL_IO, BAS$$SIGNAL_CTRLC
```

```
                                                        .EXTRN  BAS$K_OUTOF_DAT
                                                        .EXTRN  BAS$K_ENDFI[DEV
                                                        .EXTRN  BAS$K_NOTENODAT
                                                        .EXTRN  BAS$K_TOOLITDAT
                                                        .EXTRN  BAS$K_RECFILTOO

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

                                    0000 00000          .ENTRY  BAS$WAIT, Save nothing                  ; 1651
                50        04    AC  D0 00002          MOVL    TIME, R0                                ; 1693
                                    03  18 00006        BGEQ    1$                                      ;
                50              50  CE 00008          MNEGL   R0, R0                                  ;
        000000FF  8F            50  D1 0000B 1$:      CMPL    R0, #255                                ;
                                    04  15 00012        BLEQ    2$                                      ;
                50        FF    8F  9A 00014          MOVZBL  #255, R0                                ;
       00000000'  EF            50  B0 00018 2$:      MOVW    R0, WAIT                                ;
                                    04 0001F            RET                                             ; 1695

; Routine Size:  32 bytes,    Routine Base:  _BAS$CODE + 0000

;  402          1696  1
```

```
;  404          1697  1 ROUTINE BAS$$READ_WAIT                    .Read the module level OWN WAIT
;  405          1698  1     : =
;  406          1699  1
;  407          1700  1 !++
;  408          1701  1 ! FUNCTIONAL DESCRIPTION:
;  409          1702  1 !
;  410          1703  1 !       Read the module level OWN WAIT and return it.  The value of this
;  411          1704  1 !       function is the current contents of wait.  All routines that need
;  412          1705  1 !       this value must call this routine.
;  413          1706  1 !
;  414          1707  1 !
;  415          1708  1 ! FORMAL PARAMETERS:
;  416          1709  1 !
;  417          1710  1 !       NONE
;  418          1711  1 !
;  419          1712  1 ! IMPLICIT INPUTS:
;  420          1713  1 !
;  421          1714  1 !       Reads the module level OWN WAIT
;  422          1715  1 !
;  423          1716  1 ! IMPLICIT OUTPUTS:
;  424          1717  1 !
;  425          1718  1 !       None
;  426          1719  1 !
;  427          1720  1 ! ROUTINE VALUE:
;  428          1721  1 !
;  429          1722  1 !       The contents of module level OWN WAIT
;  430          1723  1 !
;  431          1724  1 ! SIDE EFFECTS:
;  432          1725  1 !
;  433          1726  1 !       None
;  434          1727  1 !--
;  435          1728  1
;  436          1729  2     BEGIN
;  437          1730  2 !+
;  438          1731  2 ! Just return the value of the module level OWN WAIT
;  439          1732  2 !-
;  440          1733  2     RETURN .WAIT;
;  441          1734  1     END;                                !End of BAS$$READ_WAIT


                                        0000 00000 BAS$$READ_WAIT:
                                                           .WORD   Save nothing                    ; 1697
                          50 00000000'  EF  3C 00002       MOVZWL  WAIT, R0                        ; 1733
                                            04 00C09       RET                                     ; 1734

; Routine Size:  10 bytes,    Routine Base:  _BAS$CODE + 0020

;  442          1735  1
```

```
;  444          1736  1 GLOBAL ROUTINE BAS$RECOUNT                        ! RECOUNT
;  445          1737  1     : =
;  446          1738  1
;  447          1739  1 !++
;  448          1740  1 ! FUNCTIONAL DESCRIPTION:
;  449          1741  1 !
;  450          1742  1 !       This routine supports the Basic RECOUNT function.  It returns the number
;  451          1743  1 !       of bytes read on the last Get.  It utilizes a piece of OWN storage which
;  452          1744  1 !       is written to by the record processing levels which do Gets.  In order
;  453          1745  1 !       to keep the OWN storage from having to be global, this routine is included
;  454          1746  1 !       in this module
;  455          1747  1 !
;  456          1748  1 ! FORMAL PARAMETERS:
;  457          1749  1 !
;  458          1750  1 !       NONE
;  459          1751  1 !
;  460          1752  1 ! IMPLICIT INPUTS:
;  461          1753  1 !
;  462          1754  1 !       RECOUNT.rl              The number of bytes read on the last GET
;  463          1755  1 !
;  464          1756  1 ! IMPLICIT OUTPUTS:
;  465          1757  1 !
;  466          1758  1 !       NONE
;  467          1759  1 !
;  468          1760  1 ! ROUTINE VALUE:
;  469          1761  1 !
;  470          1762  1 !       NUM_OF_BYTES.wl.v       number of bytes read on last Get
;  471          1763  1 !
;  472          1764  1 ! SIDE EFFECTS:
;  473          1765  1 !
;  474          1766  1 !--
;  475          1767  1
;  476          1768  2     BEGIN
;  477          1769  2     RETURN .RECOUNT
;  478          1770  1     END;                                        ! End of BAS$RECOUNT


                                        0000 00000          .ENTRY  BAS$RECOUNT, Save nothing           ; 1736
                          50 00000000'  EF  D0 00002        MOVL    RECOUNT, R0                         ; 1769
                                            04 00009        RET                                         ; 1770

; Routine Size:  10 bytes,    Routine Base:  _BAS$CODE + 002A

;  479          1771  1
```

```
;  481          1772  1 GLOBAL ROUTINE BAS$$RECOU_INIT : NOVALUE =      ! Initialize RECOUNT
;  482          1773  1
;  483          1774  1 !++
;  484          1775  1 ! FUNCTIONAL DESCRIPTION:
;  485          1776  1 !
;  486          1777  1 !
;  487          1778  1 !       This routine initializes the RECOUNT variable.  It is used before a RUN
;  488          1779  1 !       compiler command in case the previous run of the user's program left
;  489          1780  1 !       something in RECOUNT.
;  490          1781  1 !
;  491          1782  1 ! FORMAL PARAMETERS:
;  492          1783  1 !
;  493          1784  1 !       NONE
;  494          1785  1 !
;  495          1786  1 ! IMPLICIT INPUTS:
;  496          1787  1 !
;  497          1788  1 !       NONE
;  498          1789  1 !
;  499          1790  1 ! IMPLICIT OUTPUTS:
;  500          1791  1 !
;  501          1792  1 !       RECOUNT.wl      Always set to zero.
;  502          1793  1 !
;  503          1794  1 ! ROUTINE VALUE:
;  504          1795  1 !
;  505          1796  1 !       NONE
;  506          1797  1 !
;  507          1798  1 ! SIDE EFFECTS:
;  508          1799  1 !
;  509          1800  1 !--
;  510          1801  1
;  511          1802  2     BEGIN
;  512          1803  2     RECOUNT = 0;
;  513          1804  1     END;                                    ! End of BAS$$RECOU_INIT


                                        0000 00000          .ENTRY  BAS$$RECOU_INIT, Save nothing          ; 1772
                              00000000'  EF  D4 00002       CLRL    RECOUNT                                ; 1803
                                           04 00008         RET                                            ; 1804

; Routine Size:  9 bytes,    Routine Base:  _BAS$CODE + 0034

;  514          1805  1
```

```
516          1806  1 GLOBAL ROUTINE BAS$$BLNK_LINE (                    ! write a blank line
517          1807  1         FORMAT_CHAR) : CALL_CCB NOVALUE =
518          1808  1
519          1809  1 !++
520          1810  1 ! FUNCTIONAL DESCRIPTION:
521          1811  1 !
522          1812  1 !       Print out a blank line.  This is needed between arrays.
523          1813  1 !
524          1814  1 ! FORMAL PARAMETERS:
525          1815  1 !
526          1816  1 !       FORMAT_CHAR.rlu.v               the format character last used
527          1817  1 !
528          1818  1 ! IMPLICIT INPUTS:
529          1819  1 !
530          1820  1 !       NONE
531          1821  1 !
532          1822  1 ! IMPLICIT OUTPUTS:
533          1823  1 !
534          1824  1 !       NONE
535          1825  1 !
536          1826  1 ! COMPLETION CODES:
537          1827  1 !
538          1828  1 !       NONE
539          1829  1 !
540          1830  1 ! SIDE EFFECTS:
541          1831  1 !
542          1832  1 !       NONE
543          1833  1 !
544          1834  1 !--
545          1835  1
546          1836  2     BEGIN
547          1837  2
548          1838  2     EXTERNAL REGISTER
549          1839  2         CCB : REF BLOCK [, BYTE];
550          1840  2
551          1841  2     LOCAL
552          1842  2         RMS_STATUS;
553          1843  2
554          1844  2 !+
555          1845  2 ! Actually put out the blank line here.
556          1846  2 !-
557          1847  2     CCB [RAB$W_RSZ] = 0;
558          1848  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_BEG];
559          1849  2     CCB [LUB$B_BAS_VFC1] = BAS$K_LF;
560          1850  2     CCB [LUB$B_BAS_VFC2] = BAS$K_CR;
561          1851  2
562          1852  2     RMS_STATUS = $PUT (RAB = .CCB);
563          1853  2
564          1854  2     IF .RMS_STATUS EQL RMS$_CONTROLC
565          1855  2     THEN
566          1856  2         BAS$$SIGNAL_CTRLC ();
567          1857  2
568          1858  2     IF NOT .RMS_STATUS
569          1859  2     THEN
570          1860  2         PUT_ERROR (K_STOP);
571          1861  2
572          1862  2     RETURN;
```

```
;  573          1863  1     END;                                        !End of BAS$$BLNK_LINE


                                                                        .EXTRN  SYS$PUT

                                          0004 00000                    .ENTRY  BAS$$BLNK_LINE, Save R2                 ; 1806
                                    22    AB  B4 00002                  CLRW    34(CCB)                                 ; 1847
                         B0     AB      BC    AB  D0 00005              MOVL    -68(CCB), -80(CCB)                      ; 1848
                         DA     AB    8D01    8F  B0 0000A              MOVW    #36097, -38(CCB)                        ; 1849
                                              5B  DD 00010              PUSHL   CCB                                     ; 1852
                   00000000G    00           01  FB 00012              CALLS   #1, SYS$PUT                             ;
                                52           50  D0 00019              MOVL    R0, RMS_STATUS                          ;
                   00010651     8F           52  D1 0001C              CMPL    RMS_STATUS, #67153                      ; 1854
                                             07  12 00023              BNEQ    1$                                      ;
                   00000000G    00           00  FB 00025              CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 1856
                                07           52  E8 0002C 1$:          BLBS    RMS_STATUS, 2$                          ; 1858
                                             7E  D4 0002F              CLRL    -(SP)                                   ; 1860
                       0000V    CF           01  FB 00031              CALLS   #1, PUT_ERROR                           ;
                                                 04 00036 2$:          RET                                             ; 1863

; Routine Size:  55 bytes,    Routine Base:  _BAS$CODE + 003D

;  574          1864  1
```

```
1 GLOBAL ROUTINE BAS$$REC_MPR1                          ! Write Mat Print record
1     : JSB_REC1 NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Write one sequential formatted record and initialize for the next
1 !       BAS$$REC_MPR1 writes one record for 10 MAT PRINT A() and then
1 !       initializes the output buffer and returns start and end+1 of user
1 !       part of record buffer to be filled by caller.
1 !       FLR records are space padded.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LUB$V_FORM_CHAR         =1, comma or semicolon format character
1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
1 !       LUB$A_BUF_END           points to last char inserted into buffer
1 !                               by UDF level I/O.
1 !       LUB$V_FORCIBLE          Indicates a forcible device
1 !       LUB$V_OUTBUF_DR         Indicates that there is valid data in the output
1 !                               buffer
1 !       RAB$W_RSZ               Record size
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       LUB$B_BAS_VFC2          'Post' carriage control for terminal devices
1 !       LUB$A_BUF_PTR           Address of next char in user part
1 !                               of record buffer
1 !       LUB$A_BUF_END           Address of last+1 char in user part
1 !                               of record buffer
1 !       LUB$V_OUTBUF_DR         indicates valid data in the output buffer
1 !       LUB$A_BUF_BEG           Beginning of the user buffer
1 !       RAB$L_RBF               Pointer to the user record buffer.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LOCAL
2         RMS_STATUS;
2
2     !+
```

```
633          1922  2     ! If there is no format character, then set the 'pre' and 'post'
634          1923  2     ! carriage control to delimit a record.
635          1924  2     !-
636          1925  2
637          1926  2     IF NOT .CCB [LUB$V_FORM_CHAR]
638          1927  2     THEN
639          1928  3         BEGIN
640          1929  3         CCB [LUB$B_BAS_VFC1] = BAS$K_LF;
641          1930  3         CCB [LUB$B_BAS_VFC2] = BAS$K_CR;
642          1931  2         END;
643          1932  2
644          1933  2     !+
645          1934  2     ! Set recordsize to actual length of record
646          1935  2     !-
647          1936  2
648          1937  2     CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
649          1938  2
650          1939  2     !+
651          1940  2     ! Output buffer to RMS and check for errors
652          1941  2     ! If errors, SIGNAL BAS$_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
653          1942  2     !-
654          1943  2
655          1944  2     CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
656          1945  2     CCB [LUB$V_OUTBUF_DR] = 0;
657          1946  2
658          1947  2     RMS_STATUS = $PUT (RAB = .CCB);
659          1948  2
660          1949  2     IF .RMS_STATUS EQL RMS$_CONTROLC
661          1950  2     THEN
662          1951  2         BAS$$SIGNAL_CTRLC ();
663          1952  2
664          1953  2     IF NOT .RMS_STATUS
665          1954  2     THEN
666          1955  2
667          1956  2     !+
668          1957  2     ! Not OPEN or CONNECT - RMS record operation
669          1958  2     !-
670          1959  2
671          1960  2         PUT_ERROR (K_STOP);
672          1961  2
673          1962  2     !+
674          1963  2     ! Return next output buffer start and end addresses
675          1964  2     !-
676          1965  2
677          1966  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
678          1967  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
679          1968  2     RETURN;
680          1969  1     END;                                        ! End of rcutine - BAS$$REC_MPR1


                                 52  DD 00000 BAS$$REC_MPR1::
                                                 PUSHL   R2                                  ; 1865
06       FE   AB                 02  E0 00002    BBS     #2, -2(CCB), 1$                     ; 1926
         DA   AB      8D01       8F  B0 00007    MOVW    #36097, -38(CCB)                    ; 1929
```

```
                22   AB      B0   AB      BC   AB  A3 0000D 1$:    SUBW3   -68(CCB), -80(CCB), 34(CCB)          ; 1937
                             28   AB      EC   AB  D0 00014        MOVL    -20(CCB), 40(CCB)                    ; 1944
                             FE   AB           08  8A 00019        BICB2   #8, -2(CCB)                          ; 1945
                                               5B  DD 0001D        PUSHL   CCB                                  ; 1947
                      00000000G   00           01  FB 0001F        CALLS   #1, SYS$PUT                          ;
                                  52           50  D0 00026        MOVL    R0, RMS_STATUS                       ;
                      00010651    8F           52  D1 00029        CMPL    RMS_STATUS, #67153                   ; 1949
                                               07  12 00030        BNEQ    2$                                   ;
                      00000000G   00           00  FB 00032        CALLS   #0, BAS$$SIGNAL_CTRLC                ; 1951
                                  07           52  E8 00039 2$:    BLBS    RMS_STATUS, 3$                       ; 1953
                                               7E  D4 0003C        CLRL    -(SP)                                ; 1960
                          0000V   CF           01  FB 0003E        CALLS   #1, PUT_ERROR                        ;
                             B0   AB      EC   AB  D0 00043 3$:    MOVL    -20(CCB), -80(CCB)                   ; 1966
                                  50      D2   AB  3C 00048        MOVZWL  -46(CCB), R0                         ; 1967
                             B4   AB      EC BB40  9E 0004C        MOVAB   @-20(CCB)[R0], -76(CCB)              ;
                                               04  BA 00052        POPR    #^M<R2>                              ; 1969
                                                   05 00054        RSB                                          ;

; Routine Size:  85 bytes,    Routine Base:  _BAS$CODE + 0074

;  681          1970  1
```

```
1971  1 GLOBAL ROUTINE BAS$$REC_MPR9                    ! Mat Write sequential
1972  1     : JSB_REC9 NOVALUE =
1973  1
1974  1 !++
1975  1 ! FUNCTIONAL DESCRIPTION:
1976  1 !
1977  1 !       This routine does not write a record.  Presumably the MAT PRINT element
1978  1 !       transmitter took care of all of that.  Since we do not want a blank line
1979  1 !       after the array, there is no need to write anything here.
1980  1 !
1981  1 ! FORMAL PARAMETERS:
1982  1 !
1983  1 !       NONE
1984  1 !
1985  1 ! IMPLICIT INPUTS:
1986  1 !
1987  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
1988  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
1989  1 !
1990  1 ! IMPLICIT OUTPUTS:
1991  1 !
1992  1 !       LUB$A_BUF_PTR           Address of next char in user part
1993  1 !                               of record buffer
1994  1 !       LUB$A_BUF_END           Address of last+1 char in user part
1995  1 !                               of record buffer
1996  1 !
1997  1 ! ROUTINE VALUE:
1998  1 !
1999  1 !       NONE
2000  1 !
2001  1 ! SIDE EFFECTS:
2002  1 !
2003  1 !       NONE
2004  1 !--
2005  1
2006  2     BEGIN
2007  2
2008  2     EXTERNAL REGISTER
2009  2         CCB : REF BLOCK [, BYTE];
2010  2
2011  2     !+
2012  2     ! Return next output buffer start and end addresses
2013  2     !-
2014  2
2015  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
2016  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
2017  2     RETURN;
2018  1     END;                                          ! END BAS$$REC_MPR19
```

```
B0   AB       EC    AB  D0 00000 BAS$$REC_MPR9::
                                   MOVL    -20(CCB), -80(CCB)                  ; 2015
     50       D2    AB  3C 00005   MOVZWL  -46(CCB), R0                        ; 2016
B4   AB       EC BB40   9E 00009   MOVAB   @-20(CCB)[R0], -76(CCB)             ;
```

```
                                    05 0000F        RSB                                                    ; 2018

; Routine Size:  16 bytes,    Routine Base:  _BAS$CODE + 00C9

;  731          2019  1
```

```
1 GLOBAL ROUTINE BAS$$REC_WSL9                          ! Write sequential formatted
     : JSB_REC9 NOVALUE =

1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Write one sequential formatted record and initialize for the next
1 !       BAS$$REC_WSF1 (and BAS$$REC_WSL9) writes one output buffer and then
1 !       initializes the output buffer and returns start and end+1 of user
1 !       part of record buffer to be filled by caller.
1 !       FLR records are space padded.
1 !       /logical record number is incremented/.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       ISB$V_PRINT_INI         flag to indicate whether there was an element
1 !                               transmitter
1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN
1 !       LUB$A_BUF_END           points to last char inserted into buffer
1 !                               by UDF level I/O.
1 !       LUB$V_FORM_CHAR         The last element transmitter ended in a comma
1 !                               or semicolon format char.
1 !       LUB$V_FORCIBLE          Indicates a forcible device
1 !       LUB$V_OUTBUF_DR         Indicates that there is valid data in the output
1 !                               buffer
1 !       RAB$W_RSZ               Record size
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       ISB$V_PRINT_INI         reset flag
1 !       LUB$B_BAS_VFC2          'Post' carriage control for terminal devices
1 !       LUB$A_BUF_PTR           Address of next char in user part
1 !                               of record buffer
1 !       LUB$A_BUF_END           Address of last+1 char in user part
1 !                               of record buffer
1 !       LUB$V_OUTBUF_DR         indicates valid data in the output buffer
1 !       LUB$A_BUF_BEG           Beginning of the user buffer
1 !       RAB$L_RBF               Pointer to the user record buffer.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : RFF BLOCK [, BYTE];
```

```
2 LOCAL
2     RMS_STATUS;
2
2 !+
2 ! If last element ended with a format character and not a terminal device
2 ! then return to caller without writing anything.  With CR format, we must
2 ! PUT a whole record.
2 !-
2
2 IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;
2
2 !+
2 ! Set the 'post' carriage control to carriage return
2 ! if the last element transmitter had no format character following.
2 !-
2
2 IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$B_BAS_VFC2] = BAS$K_CR;
2
2 !+
2 ! Set recordsize to actual length of record
2 !-
2
2 CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
2
2 !+
2 ! Output buffer to RMS and check for errors
2 ! If errors, SIGNAL BAS$_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
2 !-
2
2 CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];
2 !+
2 ! Write something if there is something in the buffer or if there was no
2 ! element transmitter.
2 !-
2
2 IF .CCB [LUB$V_OUTBUF_DR] OR .CCB [ISB$V_PRINT_INI]
2 THEN
2
2     RMS_STATUS = $PUT (RAB = .CCB);
2
2     IF .RMS_STATUS EQL RMS$_CONTROLC
2     THEN
2         BAS$$SIGNAL_CTRLC ();
2
2     IF NOT .RMS_STATUS
2     THEN
2         PUT_ERROR (K_STOP);
2
2 CCB [LUB$V_OUTBUF_DR] = 0;
2 CCB [ISB$V_PRINT_INI] = 0;
2 !+
2 ! If there is no format character then set the 'pre' carriage control to LF
2 ! for the next record.  This is recursive I/O and the rest of the list when
2 ! we return should be written on the next line.
2 !-
2
```

```
;  847          2134  2      IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$B_BAS_VFC1] = BAS$K_LF;
;  848          2135  2
;  849          2136  2      !+
;  850          2137  2      ! Return next output buffer start and end addresses
;  851          2138  2      !-
;  852          2139  2
;  853          2140  2      CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
;  854          2141  2      CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
;  855          2142  2      RETURN;
;  856          2143  1      END;                                        ! END OF ROUTINE


                                        52  DD 00000 BAS$$REC_WSL9::
                                                         PUSHL   R2                                          ; 2020
                 0A      FE   AB        02  E1 00002     BBC     #2, -2(CCB), 1$                             ; 2087
                 66      FE   AB        06  E1 00007     BBC     #6, -2(CCB), 8$                             ;
                 05      FE   AB        02  E0 0000C     BBS     #2, -2(CCB), 2$                             ; 2094
                         DB   AB   8D   8F  90 00011 1$:  MOVB    #-115, -37(CCB)                             ;
             22  AB      B0   AB   BC   AB  A3 00016 2$:  SUBW3   -68(CCB), -80(CCB), 34(CCB)                 ; 2100
                         28   AB   BC   AB  D0 0001D      MOVL    -68(CCB), 40(CCB)                           ; 2107
                 05      FE   AB        03  E0 00022     BBS     #3, -2(CCB), 3$                             ; 2113
                 0C      97   AB        03  E1 00027     BBC     #3, -105(CCB), 4$                           ;
                                        5B  DD 0002C 3$: PUSHL   CCB                                         ; 2116
                    00000000G   00      01  FB 0002E     CALLS   #1, SYS$PUT                                 ;
                                52      50  D0 00035     MOVL    R0, RMS_STATUS                              ;
                    00010651    8F      52  D1 00038 4$: CMPL    RMS_STATUS, #67153                          ; 2118
                                        07  12 0003F     BNEQ    5$                                          ;
                    00000000G   00      00  FB 00041     CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 2120
                                07      52  E8 00048 5$: BLBS    RMS_STATUS, 6$                              ; 2122
                                        7E  D4 0004B     CLRL    -(SP)                                       ; 2124
                       0000V    CF      01  FB 0004D     CALLS   #1, PUT_ERROR                               ;
                          FE    AB      08  8A 00052 6$: BICB2   #8, -2(CCB)                                 ; 2126
                          97    AB      08  8A 00056     BICB2   #8, -105(CCB)                               ; 2127
                 04       FE    AB      02  E0 0005A     BBS     #2, -2(CCB), 7$                             ; 2134
                          DA    AB      01  90 0005F     MOVB    #1, -38(CCB)                                ;
                          B0    AB   EC    AB  D0 00063 7$: MOVL    -20(CCB), -80(CCB)                          ; 2140
                                50   D2    AB  3C 00068     MOVZWL  -46(CCB), R0                                ; 2141
                          B4    AB   EC BB40   9E 0006C     MOVAB   @-20(CCB)[R0], -76(CCB)                     ;
                                             04  BA 00072 8$: POPR    #^M<R2>                                 ; 2143
                                                 05 00074     RSB                                             ;

; Routine Size:  117 bytes,    Routine Base:  _BAS$CODE + 00D9

;  857          2144  1
```

```
1 GLOBAL ROUTINE BAS$$REC_RSL0                    ! Read initialization
1     : JSB_REC0 NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       BAS$$REC_RSF0 (and BAS$$REC_RSF1) reads one record if this is not a terminal.
1 !       Then return start and end+1 of user
1 !       part of record to be processed as input.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
1 !       LUB$V_TERM_DEV          flag in LUB which indicates a terminal device.
1 !       RAB$W_RSZ               word in the RAB which contains the buffer size.
1 !       RAB$L_RBF               longword in RAB which points to the buffer.
1 !       LUB$L_WAIT_TIME         Max time to wait for input, in seconds.
1 !       WAIT                    The module level OWN WAIT
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       RECOUNT                 Global storage to hold number of bytes read from
1 !                               last Input.
1 !       LUB$L_LOG_RECNO         Increment logical record number
1 !                               of next record to be read.
1 !       LUB$A_BUF_PTR           points to first char of user part of
1 !                               record buffer.
1 !       LUB$A_BUF_END           points to end+1 of user part of
1 !                               record buffer.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Reads next record from file on this logical unit.
1 !       Throws away things that are pending in the Print buffer for non-terminal
1 !       devices.
1 !       SIGNALs RMS errors directly.
1 !       SIGNALs BAS$K_TIMLIMEXC if a wait time was specified and we
1 !                               exceed it.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LITERAL
2         K_ESCAPE = %X'1B',
2         K_CR = %X'0D';
```

```
916     2202  2
917     2203  2     LOCAL
918     2204  2         RMS_STATUS,
919     2205  2         WAIT_TIME;                          ! Current wait time
920     2206  2 !+
921     2207  2 ! If a timeout has been specified, store information in the RAB to tell
922     2208  2 ! RMS about it.  If no timeout has been specified, clear the TMO bit
923     2209  2 ! in case there was an earlier timeout specified.
924     2210  2 !-
925     2211  2
926     2212  2 !+
927     2213  2 ! If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
928     2214  2 ! , i.e. existing EXE's can run with the LUB wait value in V2.2.
929     2215  2 !-
930     2216  2     WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
931     2217  2
932     2218  3     IF ( .WAIT_TIME EQL 0)
933     2219  2     THEN
934     2220  2         CCB [RAB$V_TMO] = 0
935     2221  2     ELSE
936     2222  3         BEGIN
937     2223  3         CCB [RAB$B_TMO] = .WAIT_TIME;
938     2224  3         CCB [RAB$V_TMO] = 1;
939     2225  2         END;
940     2226  2
941     2227  2
942     2228  2 !+
943     2229  2 ! Set the Read-no-echo RMS bit based on the user's last call to
944     2230  2 ! ECHO or NOECHO.
945     2231  2 !-
946     2232  2     CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
947     2233  2
948     2234  2     !+
949     2235  2     ! Check to see if this is a terminal device.  If this is NOT
950     2236  2     ! a terminal then do a GET.  GETs for terminals are done each time more
951     2237  2     ! data are needed.
952     2238  2     ! Read record into buffer using RMS and check for errors
953     2239  2     !-
954     2240  2
955     2241  3     IF ( NOT .CCB [LUB$V_TERM_DEV] OR .CCB [LUB$V_ANSI])
956     2242  2     THEN
957     2243  3         BEGIN
958     2244  3
959     2245  3         LOCAL
960     2246  3             TEMP_CCB : REF BLOCK [, BYTE];      ! Temporary CCB
961     2247  3         TEMP_CCB = .CCB [LUB$A_BUDDY_PTR];
962     2248  3
963     2249  3         !+
964     2250  3         ! If there is something pending in the Print buffer, then $PUT it.
965     2251  3         ! It cannot become a prompt, because RMS will throw away prompts
966     2252  3         ! to disk files; therefore we must $PUT it.
967     2253  3         !-
968     2254  3         IF (NOT .CCB [LUB$V_TERM_DEV]) AND .TEMP_CCB [LUB$V_OUTBUF_DR]
969     2255  3         THEN
970     2256  4             BEGIN
971     2257  4             TEMP_CCB [RAB$W_RSZ] = .TEMP_CCB [LUB$A_BUF_PTR] - .TEMP_CCB [LUB$A_BUF_BEG];
972     2258  4             TEMP_CCB [RAB$L_RBF] = .TEMP_CCB [LUB$A_BUF_BEG];
```

```
2259  4
2260  4            RMS_STATUS = $PUT (RAB = .TEMP_CCB);
2261  4
2262  4            IF .RMS_STATUS EQL RMS$_CONTROLC
2263  4            THEN
2264  4                BAS$$SIGNAL_CTRLC ();
2265  4
2266  4            IF NOT .RMS_STATUS
2267  4            THEN
2268  4                PUT_ERROR (K_STOP);
2269  3            END;
2270  3
2271  3        TEMP_CCB [LUB$A_BUF_PTR] = .TEMP_CCB [LUB$A_BUF_BEG];
2272  3
2273  3        RMS_STATUS = $GET (RAB = .CCB);
2274  3
2275  3        IF .RMS_STATUS EQL RMS$_CONTROLC
2276  3        THEN
2277  3            BAS$$SIGNAL_CTRLC ();
2278  3
2279  3        IF NOT .RMS_STATUS
2280  3        THEN
2281  3            GET_ERROR (K_STOP);
2282  3
2283  3        !+
2284  3        ! Set RECOUNT to the number of bytes read
2285  3        ! If the file is a terminal format file, then RECOUNT has to be
2286  3        ! adjusted for the carriage control terminator.  Because RMS does not return
2287  3        ! a terminator for a file, we unconditionally put a CRLF on the end and
2288  3        ! bump RECOUNT by 2.
2289  3        !-
2290  3
2291  5        RECOUNT = .CCB [RAB$W_RSZ] + (IF (.CCB [LUB$V_TERM_FOR]) AND ((.CCB [LUB$B_RAT] AND FAB$M_CR) NEQU 0
2292  3        THEN 2 ELSE 0);
2293  3 !+
2294  3 ! Put the CR into the STV field since RMS doesn't
2295  3 ! We should only do this if the record attributes indicate a CR format.
2296  3 !-
2297  3
2298  3        IF (.CCB [LUB$B_RAT] AND FAB$M_CR) NEQU 0 THEN CCB [RAB$L_STV] = 13;
2299  3
2300  3        !+
2301  3        ! Return start-1 and end+1 address of record just read
2302  3        !-
2303  3
2304  3        CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
2305  3        CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
2306  3        END
2307  2    ELSE
2308  2
2309  2    !+
2310  2    ! This is a terminal.  Force a no data in the buffer condition
2311  2    ! so the first GET is done on the element transmitter after the
2312  2    ! Prompt (if any) is known.
2313  2    !-
2314  2
2315  3        BEGIN
```

```
; 1030          2316  3         CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
; 1031          2317  3         CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_PTR];
; 1032          2318  2         END;
; 1033          2319  2
; 1034          2320  2     RETURN;
; 1035          2321  1     END;                                        . End of BAS$$REC_RSL0


                                                                        .EXTRN  SYS$GET

                                                 0C  BB 00000 BAS$$REC_RSL0::
                                                                        PUSHR   #^M<R2,R3>                                  ; 2145
                                50 00000000'     EF  3C 00002           MOVZWL  WAIT, R0                                    ; 2216
                                                 0C  12 00009           BNEQ    1$                                          ;
                                50        CC     AB  D0 0000B           MOVL    -52(CCB), WAIT_TIME                         ;
                                                 06  12 0000F           BNEQ    1$                                          ; 2218
                        07      AB               02  8A 00011           BICB2   #2, 7(CCB)                                  ; 2220
                                                 08  11 00015           BRB     2$                                          ;
                        1F      AB               50  90 00017 1$:       MOVB    WAIT_TIME, 31(CCB)                          ; 2223
                        07      AB               02  88 0001B           BISB2   #2, 7(CCB)                                  ; 2224
 07   AB        01              00        A0     AB  F0 0001F 2$:       INSV    -96(CCB), #0, #1, 7(CCB)                    ; 2232
           08    FE      AB                       05  E1 00026           BBC     #5, -2(CCB), 3$                             ; 2241
           03    A1      AB                       04  E0 0002B           BBS     #4, -95(CCB), 3$                            ;
                                                 00A3  31 00030           BRW     11$                                         ;
                                52        B8     AB  D0 00033 3$:       MOVL    -72(CCB), TEMP_CCB                          ; 2247
           37    FE      AB                       05  E0 00037           BBS     #5, -2(CCB), 5$                             ; 2254
           32    FE      A2                       03  E1 0003C           BBC     #3, -2(TEMP_CCB), 5$                        ;
     22    A2    B0      A2        BC             A2  A3 00041           SUBW3   -68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB)  ; 2257
           28    A2        BC                     A2  D0 00048           MOVL    -68(TEMP_CCB), 40(TEMP_CCB)                 ; 2258
                                                 52  DD 0004D           PUSHL   TEMP_CCB                                    ; 2260
     00000000G   00                              01  FB 0004F           CALLS   #1, SYS$PUT                                 ;
                 53                              50  D0 00056           MOVL    R0, RMS_STATUS                              ;
     00010651    8F                              53  D1 00059           CMPL    RMS_STATUS, #67153                          ; 2262
                                                 07  12 00060           BNEQ    4$                                          ;
     00000000G   00                              00  FB 00062           CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 2264
                 07                              53  E8 00069 4$:       BLBS    RMS_STATUS, 5$                              ; 2266
                                                 7E  D4 0006C           CLRL    -(SP)                                       ; 2268
       0000V     CF                              01  FB 0006E           CALLS   #1, PUT_ERROR                               ;
         B0      A2        BC                     A2  D0 00073 5$:       MOVL    -68(TEMP_CCB), -80(TEMP_CCB)                ; 2271
                                                 5B  DD 00078           PUSHL   CCB                                         ; 2273
     00000000G   00                              01  FB 0007A           CALLS   #1, SYS$GET                                 ;
                 53                              50  D0 00081           MOVL    R0, RMS_STATUS                              ;
     00010651    8F                              53  D1 00084           CMPL    RMS_STATUS, #67153                          ; 2275
                                                 07  12 0008B           BNEQ    6$                                          ;
     00000000G   00                              00  FB 0008D           CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 2277
                 07                              53  E8 00094 6$:       BLBS    RMS_STATUS, 7$                              ; 2279
                                                 7E  D4 00097           CLRL    -(SP)                                       ; 2281
       0000V     CF                              01  FB 00099           CALLS   #1, GET_ERROR                               ;
           0A    FE      AB                       04  E1 0009E 7$:       BBC     #4, -2(CCB), 8$                             ; 2291
           05    F6      AB                       01  E1 000A3           BBC     #1, -10(CCB), 8$                            ;
                 50                               02  D0 000A8           MOVL    #2, R0                                      ;
                                                  02  11 000AB           BRB     9$                                          ;
                                                  50  D4 000AD 8$:       CLRL    R0                                          ;
                 51        22                     AB  3C 000AF 9$:       MOVZWL  34(CCB), R1                                 ;
00000000'  EF    50                               51  C1 000B3           ADDL3   R1, R0, RECOUNT                             ;
           04    F6      AB                       01  E1 000BB           BBC     #1, -10(CCB), 10$                           ; 2298
```

```
                    0C   AB          0D  D0 000C0          MOVL    #13, 12(CCB)                          ;
          B0   AB   28   AB          01  C3 000C4 10$:    SUBL3   #1, 40(CCB), -80(CCB)                 ; 2304
                     50       22    AB  3C 000CA          MOVZWL  34(CCB), R0                           ; 2305
                    B4   AB   28 BB40  9E 000CE          MOVAB   @40(CCB)[R0], -76(CCB)                ;
                               0A  11 000D4              BRB     12$                                   ; 2241
                    B0   AB   28   AB  D0 000D6 11$:     MOVL    40(CCB), -80(CCB)                     ; 2316
                    B4   AB   B0   AB  D0 000DB          MOVL    -80(CCB), -76(CCB)                    ; 2317
                               0C  BA 000E0 12$:         POPR    #^M<R2,R3>                            ; 2321
                                   05 000E2              RSB                                           ;

; Routine Size:  227 bytes,    Routine Base:  _BAS$CODE + 014E

; 1036          2322  1
```

```
1 GLOBAL ROUTINE BAS$$REC_MINO                          ! MAT Input initialization
1     : JSB_RECO NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       BAS$$REC_RSF0 (and BAS$$REC_RSF1) reads one record if this is not a terminal.
1 !       Then return start and end+1 of user
1 !       part of record to be processed as input.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
1 !       LUB$V_TERM_DEV          flag in LUB which indicates a terminal device.
1 !       RAB$W_RSZ               word in the RAB which contains the buffer size.
1 !       RAB$L_RBF               longword in RAB which points to the buffer.
1 !       LUB$L_WAIT_TIME         Max time to wait for input, in seconds.
1 !       WAIT                    Module level OWN WAIT
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       RECOUNT                 Global storage to hold number of bytes read from
1 !                               last Input.
1 !       LUB$L_LOG_RECNO         Increment logical record number
1 !                               of next record to be read.
1 !       LUB$A_BUF_PTR           points to first char of user part of
1 !                               record buffer.
1 !       LUB$A_BUF_END           points to end+1 of user part of
1 !                               record buffer.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Reads next record from file on this logical unit.
1 !       Throws away things that are pending in the Print buffer for non-terminal
1 !       devices.
1 !       SIGNALs BAS$K_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
1 !       SIGNALs BAS$K_ENDFILDEV (11='END-OF-FILE ON DEVICE')
1 !       SIGNALs BAS$K_RECFILTOO if record too big
1 !       SIGNALs BAS$K_TIMLIMEXC if a wait time was specified and we
1 !                               exceed it.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LITERAL
```

```
; 1095          2380  2          K_ESCAPE = %X'1B',
; 1096          2381  2          K_CR = %X'0D';
; 1097          2382  2
; 1098          2383  2      LOCAL
; 1099          2384  2          RMS_STATUS,
; 1100          2385  2          WAIT_TIME;                          !Current wait time
; 1101          2386  2  !+
; 1102          2387  2  ! If a timeout has been specified, store information in the RAB to tell
; 1103          2388  2  ! RMS about it.  If no timeout has been specified, clear the TMO bit
; 1104          2389  2  ! in case there was an earlier timeout specified.
; 1105          2390  2  !-
; 1106          2391  2
; 1107          2392  2
; 1108          2393  2  !+
; 1109          2394  2  ! If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
; 1110          2395  2  ! , i.e. existing EXE's can run with the LUB wait value in V2.2.
; 1111          2396  2  !-
; 1112          2397  2      WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
; 1113          2398  2
; 1114          2399  3      IF (.WAIT_TIME EQL 0)
; 1115          2400  2      THEN
; 1116          2401  2          CCB [RAB$V_TMO] = 0
; 1117          2402  2      ELSE
; 1118          2403  3          BEGIN
; 1119          2404  3          CCB [RAB$B_TMO] = .WAIT_TIME;
; 1120          2405  3          CCB [RAB$V_TMO] = 1;
; 1121          2406  2          END;
; 1122          2407  2
; 1123          2408  2
; 1124          2409  2  !+
; 1125          2410  2  ! Set the Read-no-echo RMS bit based on the user's last call to
; 1126          2411  2  ! ECHO or NOECHO.
; 1127          2412  2  !-
; 1128          2413  2      CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
; 1129          2414  2
; 1130          2415  2      !+
; 1131          2416  2      ! Check to see if this is a terminal device.  If this is NOT
; 1132          2417  2      ! a terminal then do a GET.  GETs for terminals are done each time more
; 1133          2418  2      ! data are needed.
; 1134          2419  2      ! Read record into buffer using RMS and check for errors
; 1135          2420  2      ! If end-of-file, SIGNAL BAS$K_ENDFILDEV (11='END-OF-FILE ON DEVICE')
; 1136          2421  2      ! If record too big for record buffer, SIGNAL BAS$K_RECFILTOO.
; 1137          2422  2      ! If errors, SIGNAL BAS$K_FATSYSIO (12='FATAL SYSTEM I/O ERROR')
; 1138          2423  2      !-
; 1139          2424  2
; 1140          2425  3      IF ( NOT .CCB [LUB$V_TERM_DEV])
; 1141          2426  2      THEN
; 1142          2427  3          BEGIN
; 1143          2428  3
; 1144          2429  3          LOCAL
; 1145          2430  3              TEMP_CCB : REF BLOCK [, BYTE];      ! Temporary CCB
; 1146          2431  3          TEMP_CCB = .CCB [LUB$A_BUDDY_PTR];
; 1147          2432  3
; 1148          2433  3          !+
; 1149          2434  3          ! If there is something pending in the Print buffer, then $PUT it.
; 1150          2435  3          ! It cannot become a prompt, because RMS will throw away prompts
; 1151          2436  3          ! to disk files; therefore we must $PUT it.
```

```
; 1152          2437  3          !-
; 1153          2438  3          IF (NOT .CCB [LUB$V_TERM_DEV]) AND .TEMP_CCB [LUB$V_OUTBUF_DR]
; 1154          2439  3          THEN
; 1155          2440  4              BEGIN
; 1156          2441  4              TEMP_CCB [RAB$W_RSZ] = .TEMP_CCB [LUB$A_BUF_PTR] - .TEMP_CCB [LUB$A_BUF_BEG];
; 1157          2442  4              TEMP_CCB [RAB$L_RBF] = .TEMP_CCB [LUB$A_BUF_BEG];
; 1158          2443  4
; 1159          2444  4              RMS_STATUS = $PUT (RAB = .CCB);
; 1160          2445  4
; 1161          2446  4              IF .RMS_STATUS EQL RMS$_CONTROLC
; 1162          2447  4              THEN
; 1163          2448  4                  BAS$$SIGNAL_CTRLC ();
; 1164          2449  4
; 1165          2450  4              IF NOT .RMS_STATUS
; 1166          2451  4              THEN
; 1167          2452  4                  PUT_ERROR (K_STOP);
; 1168          2453  3              END;
; 1169          2454  3
; 1170          2455  3          TEMP_CCB [LUB$A_BUF_PTR] = .TEMP_CCB [LUB$A_BUF_BEG];
; 1171          2456  3
; 1172          2457  3          RMS_STATUS = $GET (RAB = .CCB);
; 1173          2458  3
; 1174          2459  3          IF .RMS_STATUS EQL RMS$_CONTROLC
; 1175          2460  3          THEN
; 1176          2461  3              BAS$$SIGNAL_CTRLC ();
; 1177          2462  3
; 1178          2463  3          IF NOT .RMS_STATUS
; 1179          2464  3          THEN
; 1180          2465  3              GET_ERROR (K_STOP);
; 1181          2466  3
; 1182          2467  3          !+
; 1183          2468  3          ! Set RECOUNT to the number of bytes read
; 1184          2469  3          ! If the file is a terminal format file, then RECOUNT has to be
; 1185          2470  3          ! adjusted for the carriage control terminator.
; 1186          2471  3          !-
; 1187          2472  3
; 1188          2473  4          RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
; 1189          2474  4                  SET
; 1190          2475  4                  [K_ESCAPE] : .CCB [RAB$W_STV2];
; 1191          2476  4                  [K_CR] : 2;
; 1192          2477  4                  [OTHERWISE] : 0;
; 1193          2478  3                  TES ELSE 0);
; 1194          2479  3
; 1195          2480  3          !+
; 1196          2481  3          ! Return start-1 and end+1 address of record just read
; 1197          2482  3          !-
; 1198          2483  3
; 1199          2484  3          CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
; 1200          2485  3          CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
; 1201          2486  3
; 1202          2487  3          !+
; 1203          2488  3          ! Check for an '&' as the last character of the record.  If it is there,
; 1204          2489  3          ! it is a continuation character and signifies that there is more data to
; 1205          2490  3          ! come in the next record.
; 1206          2491  3          !-
; 1207          2492  3
; 1208          2493  3          IF .(.CCB [LUB$A_BUF_END] - 1)<0, 8> EQLU K_MAT_CONT_CHAR
```

```
; 1209          2494  3         THEN
; 1210          2495  4             BEGIN
; 1211          2496  4             CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_END] - 1;
; 1212          2497  4             CCB [ISB$V_MAT_CONT] = 1;
; 1213          2498  4             END
; 1214          2499  3         ELSE
; 1215          2500  3             CCB [ISB$V_MAT_CONT] = 0;
; 1216          2501  3
; 1217          2502  3         END
; 1218          2503  2     ELSE
; 1219          2504  2
; 1220          2505  2     !+
; 1221          2506  2     ! This is a terminal.  Force a no data in the buffer condition
; 1222          2507  2     ! so the first GET is done on the element transmitter after the
; 1223          2508  2     ! Prompt (if any) is known.  Set the MAT Input continuation flag so that the element
; 1224          2509  2     ! transmitter (REC1) can read the first record.
; 1225          2510  2     !-
; 1226          2511  2
; 1227          2512  3         BEGIN
; 1228          2513  3         CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF];
; 1229          2514  3         CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_PTR];
; 1230          2515  3         CCB [ISB$V_MAT_CONT] = 1;
; 1231          2516  2         END;
; 1232          2517  2
; 1233          2518  2     RETURN;
; 1234          2519  1     END;                                    ! End of BAS$$REC_MIN0


                                        0C  BB 00000 BAS$$REC_MIN0::
                                                     PUSHR   #^M<R2,R3>                                          ; 2323
                     50 00000000'       EF  3C 00002         MOVZWL  WAIT, R0                                    ; 2397
                                        0C  12 00009         BNEQ    1$
                     50        CC       AB  D0 0000B         MOVL    -52(CCB), WAIT_TIME
                                        06  12 0000F         BNEQ    1$                                          ; 2399
                 07  AB                 02  8A 00011         BICB2   #2, 7(CCB)                                  ; 2401
                                        08  11 00015         BRB     2$
                 1F  AB                 50  90 00017 1$:     MOVB    WAIT_TIME, 31(CCB)                          ; 2404
                 07  AB                 02  88 0001B         BISB2   #2, 7(CCB)                                  ; 2405
07  AB       01      00        A0       AB  F0 0001F 2$:     INSV    -96(CCB), #0, #1, 7(CCB)                    ; 2413
             03  FE  AB                 05  E1 00026         BBC     #5, -2(CCB), 3$                             ; 2425
                                    00BC  31 0002B         BRW     12$
                     52        B8       AB  D0 0002E 3$:     MOVL    -72(CCB), TEMP_CCB                          ; 2431
             37  FE  AB                 05  E0 00032         BBS     #5, -2(CCB), 5$                             ; 2438
             32  FE  A2                 03  E1 00037         BBC     #3, -2(TEMP_CCB), 5$
         22  A2  B0  A2        BC       A2  A3 0003C         SUBW3   -68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB)  ; 2441
             28  A2  BC       A2        D0 00043             MOVL    -68(TEMP_CCB), 40(TEMP_CCB)                 ; 2442
                                        5B  DD 00048         PUSHL   CCB                                         ; 2444
         00000000G  00                  01  FB 0004A         CALLS   #1, SYS$PUT
                    53                  50  D0 00051         MOVL    R0, RMS_STATUS
         00010651   8F                  53  D1 00054         CMPL    RMS_STATUS, #67153                          ; 2446
                                        07  12 0005B         BNEQ    4$
         00000000G  00                  00  FB 0005D         CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 2448
                    07                  53  E8 00064 4$:     BLBS    RMS_STATUS, 5$                              ; 2450
                                        7E  D4 00067         CLRL    -(SP)                                       ; 2452
```

```
                    0000V  CF          01  FB 00069          CALLS   #1, PUT_ERROR                          ;
                      B0   A2     BC   A2  D0 0006E 5$:      MOVL    -68(TEMP_CCB), -80(TEMP_CCB)           ; 2455
                                       5B  DD 00073          PUSHL   CCB                                    ; 2457
                00000000G  00          01  FB 00075          CALLS   #1, SYS$GET                            ;
                           53          50  D0 0007C          MOVL    R0, RMS_STATUS                         ;
                00010651   8F          53  D1 0007F          CMPL    RMS_STATUS, #67153                     ; 2459
                                       07  12 00086          BNEQ    6$                                     ;
                00000000G  00          00  FB 00088          CALLS   #0, BAS$$SIGNAL_CTRLC                  ; 2461
                           07          53  E8 0008F 6$:      BLBS    RMS_STATUS, 7$                         ; 2463
                                       7E  D4 00092          CLRL    -(SP)                                  ; 2465
                    0000V  CF          01  FB 00094          CALLS   #1, GET_ERROR                          ;
             19       FE  AB          04  E1 00099 7$:      BBC     #4, -2(CCB), 9$                        ; 2473
                         50     0C   AB  3C 0009E            MOVZWL  12(CCB), R0                            ;
                         1B          50  B1 000A2            CMPW    R0, #27                                ; 2475
                                     06  12 000A5            BNEQ    8$                                     ;
                         50     0E   AB  3C 000A7            MOVZWL  14(CCB), R0                            ;
                                     0C  11 000AB            BRB     10$                                    ;
                         0D          50  B1 000AD 8$:       CMPW    R0, #13                                ; 2476
                                     05  12 000B0            BNEQ    9$                                     ;
                         50          02  D0 000B2            MOVL    #2, R0                                 ;
                                     02  11 000B5            BRB     10$                                    ;
                                     50  D4 000B7 9$:        CLRL    R0                                     ; 2473
                         51     22   AB  3C 000B9 10$:       MOVZWL  34(CCB), R1                            ;
        00000000'  EF    50          51  C1 000BD            ADDL3   R1, R0, RECOUNT                        ;
              B0   AB       28  AB     01  C3 000C5            SUBL3   #1, 40(CCB), -80(CCB)                  ; 2484
                         50     22   AB  3C 000CB            MOVZWL  34(CCB), R0                            ; 2485
                       B4  AB     28 BB40  9E 000CF            MOVAB   @40(CCB)[R0], -76(CCB)                 ;
                         50     B4   AB  D0 000D5            MOVL    -76(CCB), R0                           ; 2493
                         26     FF   A0  91 000D9            CMPB    -1(R0), #38                            ;
                                     05  12 000DD            BNEQ    11$                                    ;
                                B4   AB  D7 000DF            DECL    -76(CCB)                               ; 2496
                                     10  11 000E2            BRB     13$                                    ; 2497
                       97  AB        02  8A 000E4 11$:       BICB2   #2, -105(CCB)                          ; 2500
                                     0E  11 000E8            BRB     14$                                    ; 2425
                       B0  AB     28  AB  D0 000EA 12$:       MOVL    40(CCB), -80(CCB)                      ; 2513
                       B4  AB     B0  AB  D0 000EF            MOVL    -80(CCB), -76(CCB)                     ; 2514
                       97  AB        02  88 000F4 13$:       BISB2   #2, -105(CCB)                          ; 2515
                                     0C  BA 000F8 14$:       POPR    #^M<R2,R3>                             ; 2519
                                         05 000FA            RSB                                            ;

; Routine Size:  251 bytes,    Routine Base:  _BAS$CODE + 0231

; 1235          2520  1
```

```
; 1237          2521  1 GLOBAL ROUTINE BAS$$REC_RSL1                    ! Read element transmitter
; 1238          2522  1     : JSB_REC1 =
; 1239          2523  1
; 1240          2524  1 !++
; 1241          2525  1 ! FUNCTIONAL DESCRIPTION:
; 1242          2526  1 !
; 1243          2527  1 !       BAS$$REC_RSL1 reads one record if this is a terminal device.
; 1244          2528  1 !       Otherwise an error is signalled.
; 1245          2529  1 !       Then return start and end+1 of user
; 1246          2530  1 !       part of record to be processed as input.
; 1247          2531  1 !
; 1248          2532  1 ! FORMAL PARAMETERS:
; 1249          2533  1 !
; 1250          2534  1 !       NONE
; 1251          2535  1 !
; 1252          2536  1 ! IMPLICIT INPUTS:
; 1253          2537  1 !
; 1254          2538  1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
; 1255          2539  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
; 1256          2540  1 !       LUB$V_TERM_DEV          flag indicating a terminal device.
; 1257          2541  1 !       RAB$L_RBF               Pointer to buffer
; 1258          2542  1 !       RAB$W_RSZ               buffer size
; 1259          2543  1 !
; 1260          2544  1 ! IMPLICIT OUTPUTS:
; 1261          2545  1 !
; 1262          2546  1 !       RECOUNT                 Own storage for RECOUNT function.
; 1263          2547  1 !       LUB$A_BUF_PTR           points to first char of user part of
; 1264          2548  1 !                               record buffer.
; 1265          2549  1 !       LUB$A_BUF_END           points to end+1 of user part of
; 1266          2550  1 !                               record buffer.
; 1267          2551  1 !
; 1268          2552  1 ! ROUTINE VALUE:
; 1269          2553  1 !
; 1270          2554  1 !       NONE
; 1271          2555  1 !
; 1272          2556  1 ! SIDE EFFECTS:
; 1273          2557  1 !
; 1274          2558  1 !       Reads next record from file on this logical unit.
; 1275          2559  1 !       SIGNALs Insufficient data or any resultant RMS errors.
; 1276          2560  1 !--
; 1277          2561  1
; 1278          2562  2     BEGIN
; 1279          2563  2
; 1280          2564  2     EXTERNAL REGISTER
; 1281          2565  2         CCB : REF BLOCK [, BYTE];
; 1282          2566  2
; 1283          2567  2     LOCAL
; 1284          2568  2         RMS_STATUS,
; 1285          2569  2         T_CCB : REF BLOCK [, BYTE];
; 1286          2570  2
; 1287          2571  2     LITERAL
; 1288          2572  2         K_ESCAPE = %X'1B',
; 1289          2573  2         K_CR = %X'0D';
; 1290          2574  2
; 1291          2575  2     !+
; 1292          2576  2     ! Check to see if this is a terminal device.  If this is
; 1293          2577  2     ! a terminal then do a GET.  GETs for terminals are done each time more
```

```
; 1294          2578  2     ! data are needed.  If this is not a terminal device then error.
; 1295          2579  2     ! Read record into buffer using RMS and check for errors
; 1296          2580  2     !-
; 1297          2581  2
; 1298          2582  2     IF (NOT .CCB [LUB$V_ANSI]) AND .CCB [LUB$V_TERM_DEV]
; 1299          2583  2     THEN
; 1300          2584  3         BEGIN
; 1301          2585  3
; 1302          2586  3         RMS_STATUS = $GET (RAB = .CCB);
; 1303          2587  3
; 1304          2588  3         IF .RMS_STATUS EQL RMS$_CONTROLC
; 1305          2589  3         THEN
; 1306          2590  3             BAS$$SIGNAL_CTRLC ();
; 1307          2591  3
; 1308          2592  3         IF NOT .RMS_STATUS
; 1309          2593  3         THEN
; 1310          2594  3             GET_ERROR (K_STOP);
; 1311          2595  3
; 1312          2596  3         !+
; 1313          2597  3         ! Return start-1 and end+1 address of record just read
; 1314          2598  3         ! LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
; 1315          2599  3         ! Input.  This is seen as a solution to the problem of the user entering
; 1316          2600  3         ! <return> as the response to a prompt (null input record) and an empty
; 1317          2601  3         ! or depleted buffer which requires another Get.
; 1318          2602  3         ! The algorithm:
; 1319          2603  3         ! 1)    Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1320          2604  3         !       T: The buffer is depleted - another Get is required.
; 1321          2605  3         ! 2)    Add one to LUB$A_BUF_PTR
; 1322          2606  3         ! 3)    Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1323          2607  3         !       T: Return the default value.
; 1324          2608  3         ! 4)    Scan for the next field.
; 1325          2609  3         !
; 1326          2610  3         !-
; 1327          2611  3
; 1328          2612  3         CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
; 1329          2613  3         CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
; 1330          2614  3         END
; 1331          2615  2     ELSE
; 1332          2616  2
; 1333          2617  2     !+
; 1334          2618  2     ! This is not a terminal device
; 1335          2619  2     ! Signal insufficient data unless this is an ANSI INPUT.
; 1336          2620  2     ! ANSI INPUT errors should cause the statement to be restarted.
; 1337          2621  2     ! (This happens in BAS$$HANDLER).
; 1338          2622  2     !-
; 1339          2623  2
; 1340          2624  2         IF NOT .CCB [LUB$V_ANSI]
; 1341          2625  2         THEN
; 1342          2626  2             BAS$$SIGNAL (BAS$K_NOTENODAT)
; 1343          2627  2         ELSE
; 1344          2628  2             BAS$$SIGNAL_IO (BAS$K_TOOLITDAT);
; 1345          2629  2
; 1346          2630  2 !+
; 1347          2631  2 ! Update the cursor position if this input was terminated by an escape.
; 1348          2632  2 ! Save cursor position if last PRINT terminator was a semi or comma.
; 1349          2633  2 ! Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
; 1350          2634  2 !-
```

```
; 1351   2635  2     T_CCB = .CCB [LUB$A_BUDDY_PTR];
; 1352   2636  3    T_CCB [LUB$L_PRINT_POS] = (IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
; 1353   2637  3                               THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
; 1354   2638  2                               ELSE 0);
; 1355   2639  2
; 1356   2640  2     !+
; 1357   2641  2     ! Set RECOUNT to the number of bytes read
; 1358   2642  2     ! If the file is a terminal format file, then RECOUNT has to be
; 1359   2643  2     ! adjusted for the carriage control terminator.
; 1360   2644  2     !-
; 1361   2645  2
; 1362   2646  3     RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
; 1363   2647  3             SET
; 1364   2648  3             [K_ESCAPE] : .CCB [RAB$W_STV2];
; 1365   2649  3             [K_CR] : 2;
; 1366   2650  3             [OTHERWISE] : 0;
; 1367   2651  2             TES ELSE 0);
; 1368   2652  2     RETURN 1;
; 1369   2653  1     END;                                        ! End of BAS$$REC_RSL1


                                 52  DD 00000 BAS$$REC_RSL1::
                                                 PUSHL   R2                                  ; 2521
   4F       A1   AB              04  E0 00002    BBS     #4, -95(CCB), 4$                    ; 2582
   38       FE   AB              05  E1 00007    BBC     #5, -2(CCB), 3$                     ;
                                 5B  DD 0000C    PUSHL   CCB                                 ; 2586
      00000000G  00              01  FB 0000E    CALLS   #1, SYS$GET                         ;
                 52              50  D0 00015    MOVL    R0, RMS_STATUS                      ;
      00010651   8F              52  D1 00018    CMPL    RMS_STATUS, #67153                  ; 2588
                                 07  12 0001F    BNEQ    1$                                  ;
      00000000G  00              00  FB 00021    CALLS   #0, BAS$$SIGNAL_CTRLC               ; 2590
                 07              52  E8 00028 1$:  BLBS  RMS_STATUS, 2$                      ; 2592
                                 7E  D4 0002B    CLRL    -(SP)                               ; 2594
            0000V  CF            01  FB 0002D    CALLS   #1, GET_ERROR                       ;
B0   AB       28  AB              01  C3 00032 2$:  SUBL3 #1, 40(CCB), -80(CCB)               ; 2612
                 50        22   AB  3C 00038    MOVZWL  34(CCB), R0                         ; 2613
              B4  AB        28 BB40  9E 0003C    MOVAB   @40(CCB)[R0], -76(CCB)              ;
                                 1D  11 00042    BRB     5$                                  ; 2582
   0D       A1   AB              04  E0 00044 3$:  BBS   #4, -95(CCB), 4$                    ; 2624
                 7E        00G  8F  9A 00049    MOVZBL  #BAS$K_NOTENODAT, -(SP)             ; 2626
      00000000G  00              01  FB 0004D    CALLS   #1, BAS$$SIGNAL                     ;
                                 0B  11 00054    BRB     5$                                  ;
                 7E        00G  8F  9A 00056 4$:  MOVZBL  #BAS$K_TOOLITDAT, -(SP)             ; 2628
      00000000G  00              01  FB 0005A    CALLS   #1, BAS$$SIGNAL_IO                  ;
                 51        B8   AB  D0 00061 5$:  MOVL    -72(CCB), T_CCB                     ; 2635
                 52        0C   AB  3C 00065    MOVZWL  12(CCB), R2                         ; 2636
                 1B              52  B1 00069    CMPW    R2, #27                             ;
                                 11  12 0006C    BNEQ    6$                                  ;
   0C       FE   A1              02  E1 0006E    BBC     #2, -2(T_CCB), 6$                   ;
                 50        22   AB  3C 00073    MOVZWL  34(CCB), R0                         ; 2637
                 50        C8   A1  C0 00077    ADDL2   -56(T_CCB), R0                      ;
                                 50  D6 0007B    INCL    R0                                  ;
                                 02  11 0007D    BRB     7$                                  ;
                                 50  D4 0007F 6$:  CLRL  R0                                  ; 2636
```

```
                        C8   A1          50  D0 00081 7$:     MOVL    R0, -56(T_CCB)                          ;
                 15     FE   AB          04  E1 00085         BBC     #4, -2(CCB), 9$                         ; 2646
                              1B          52  B1 0008A         CMPW    R2, #27                                 ; 2648
                                          06  12 0008D         BNEQ    8$                                      ;
                              50     0E   AB  3C 0008F         MOVZWL  14(CCB), R0                             ;
                                          0C  11 00093         BRB     10$                                     ;
                              0D          52  B1 00095 8$:     CMPW    R2, #13                                 ; 2649
                                          05  12 00098         BNEQ    9$                                      ;
                              50          02  D0 0009A         MOVL    #2, R0                                  ;
                                          02  11 0009D         BRB     10$                                     ;
                                          50  D4 0009F 9$:     CLRL    R0                                      ; 2646
                              51     22   AB  3C 000A1 10$:    MOVZWL  34(CCB), R1                             ;
        00000000'  EF         50          51   1 ^00A5         ADDL3   R1, R0, RECOUNT                         ;
                              50          01  DU 000AD         MOVL    #1, R0                                  ; 2652
                                          04  BA 000B0         POPR    #^M<R2>                                 ; 2653
                                              05 000B2         RSB                                             ;

; Routine Size:  179 bytes,    Routine Base:  _BAS$CODE + 032C

; 1370          2654  1
```

```
1 GLOBAL ROUTINE BAS$$REC_MIN1                          !  MAT Input element transmitter
     : JSB_REC1 =

1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       BAS$$REC_MIN1 reads one record and checks for a continuation character.
1 !       Then return start and end+1 of user
1 !       part of record to be processed as input.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
1 !       LUB$V_TERM_DEV          flag indicating a terminal device.
1 !       RAB$L_RBF               Pointer to buffer
1 !       RAB$W_RSZ               buffer size
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PECOUNT                 Own storage for RECOUNT function.
1 !       LUB$A_BUF_PTR           points to first char of user part of
1 !                               record buffer.
1 !       LUB$A_BUF_END           points to end+1 of user part of
1 !                               record buffer.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Reads next record from file on this logical unit.
1 !       SIGNALs any resultant RMS errors.
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LITERAL
2         K_ESCAPE = %X'1B',
2         K_CR = %X'0D';
2
2     LOCAL
2         RMS_STATUS,
2         T_CCB : REF BLOCK [, BYTE],
2         STATUS;                                       ! Return status to UDF of whether
2                                                       ! to keep reading
2
2     !+
```

```
; 1429          2712  2      ! Read record into buffer using RMS and check for errors and a continuation character
; 1430          2713  2      ! Signal any RMS errors directly.
; 1431          2714  2      !-
; 1432          2715  2
; 1433          2716  2      IF .CCB [ISB$V_MAT_CONT]
; 1434          2717  2      THEN
; 1435          2718  3          BEGIN
; 1436          2719  3
; 1437          2720  3          RMS_STATUS = $GET (RAB = .CCB);
; 1438          2721  3
; 1439          2722  3          IF .RMS_STATUS EQL RMS$_CONTROLC
; 1440          2723  3          THEN
; 1441          2724  3              BAS$$SIGNAL_CTRLC ();
; 1442          2725  3
; 1443          2726  3          IF NOT .RMS_STATUS
; 1444          2727  3          THEN
; 1445          2728  3              GET_ERROR (K_STOP);
; 1446          2729  3
; 1447          2730  3          !+
; 1448          2731  3          ! Return start-1 and end+1 address of record just read
; 1449          2732  3          ! LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
; 1450          2733  3          ! Input.  This is seen as a solution to the problem of the user entering
; 1451          2734  3          ! <return> as the response to a prompt (null input record) and an empty
; 1452          2735  3          ! or depleted buffer which requires another Get.
; 1453          2736  3          ! The algorithm:
; 1454          2737  3          ! 1)     Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1455          2738  3          !        T: The buffer is depleted - another Get is required.
; 1456          2739  3          ! 2)     Add one to LUB$A_BUF_PTR
; 1457          2740  3          ! 3)     Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1458          2741  3          !        T: Return the default value.
; 1459          2742  3          ! 4)     Scan for the next field.
; 1460          2743  3          !
; 1461          2744  3          !-
; 1462          2745  3
; 1463          2746  3          CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
; 1464          2747  3          CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
; 1465          2748  3
; 1466          2749  3          !+
; 1467          2750  3          ! Check for an '&' as the last character of the record.  If it is there,
; 1468          2751  3          ! it is a continuation character and signifies that there is more data to
; 1469          2752  3          ! come in the next record.
; 1470          2753  3          !-
; 1471          2754  3
; 1472          2755  3          IF .(.CCB [LUB$A_BUF_END] - 1)<0, 8> EQLU K_MAT_CONT_CHAR
; 1473          2756  3          THEN
; 1474          2757  4              BEGIN
; 1475          2758  4              CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_END] - 1;
; 1476          2759  4              CCB [ISB$V_MAT_CONT] = 1;
; 1477          2760  4              END
; 1478          2761  3          ELSE
; 1479          2762  3              CCB [ISB$V_MAT_CONT] = 0;
; 1480          2763  3
; 1481          2764  3  !+
; 1482          2765  3  ! Update the cursor position if this input was terminated by an escape.
; 1483          2766  3  ! Save the cursor position if last PRINT terminator was a semi or comma.
; 1484          2767  3  ! Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
; 1485          2768  3  !-
```

```
; 1486          2769  3         T_CCB = .CCB [LUB$A_BUDDY_PTR];
; 1487          2770  4         T_CCB [LUB$L_PRINT_POS] = (IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
; 1488          2771  4                                     THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
; 1489          2772  3                                     ELSE 0);
; 1490          2773  3
; 1491          2774  3         !+
; 1492          2775  3         ! Set RECOUNT to the number of bytes read
; 1493          2776  3         ! If the file is a terminal format file, then RECOUNT has to be
; 1494          2777  3         ! adjusted for the carriage control terminator.
; 1495          2778  3         !-
; 1496          2779  3
; 1497          2780  4         RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_ ERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
; 1498          2781  4                 SET
; 1499          2782  4                 [K_ESCAPE] : .CCB [RAB$W_STV2];
; 1500          2783  4                 [K_CR] : 2;
; 1501          2784  4                 [OTHERWISE] : 0;
; 1502          2785  3                 TES ELSE 0);
; 1503          2786  3         STATUS = 1;
; 1504          2787  3         END
; 1505          2788  2     ELSE
; 1506          2789  2         STATUS = 0;
; 1507          2790  2
; 1508          2791  2     RETURN .STATUS;
; 1509          2792  1     END;                                    ! End of BAS$$REC_MIN1


                                    52  DD 00000 BAS$$REC_MIN1::
                                                     PUSHL   R2                                  ; 2655
      03      97  AB                01  E0 00002     BBS     #1, -105(CCB), 1$                   ; 2716
                                 009E  31 00007     BRW     11$                                 ;
                                    5B  DD 0000A 1$: PUSHL   CCB                                 ; 2720
         00000000G  00              01  FB 0000C     CALLS   #1, SYS$GET                         ;
                    52              50  D0 00013     MOVL    R0, RMS_STATUS                      ;
         00010651   8F              52  D1 00016     CMPL    RMS_STATUS, #67153                  ; 2722
                                    07  12 0001D     BNEQ    2$                                  ;
         00000000G  00              00  FB 0001F     CALLS   #0, BAS$$SIGNAL_CTRLC               ; 2724
                    07              52  E8 00026 2$: BLBS    RMS_STATUS, 3$                      ; 2726
                                    7E  D4 00029     CLRL    -(SP)                               ; 2728
            0000V   CF              01  FB 0002B     CALLS   #1, GET_ERROR                       ;
B0    AB      28   AB              01  C3 00030 3$: SUBL3   #1, 40(CCB), -80(CCB)               ; 2746
                   50        22    AB  3C 00036     MOVZWL  34(CCB), R0                         ; 2747
              B4   AB        28 BB40  9E 0003A     MOVAB   @40(CCB)[R0], -76(CCB)              ;
                   50        B4    AB  D0 00040     MOVL    -76(CCB), R0                        ; 2755
                   26        FF    A0  91 00044     CMPB    -1(R0), #38                         ;
                                    09  12 00048     BNEQ    4$                                  ;
                             B4    AB  D7 0004A     DECL    -76(CCB)                            ; 2758
              97   AB              02  88 0004D     BISB2   #2, -105(CCB)                       ; 2759
                                    04  11 00051     BRB     5$                                  ; 2755
              97   AB              02  8A 00053 4$: BICB2   #2, -105(CCB)                       ; 2762
                   50        B8    AB  D0 00057 5$: MOVL    -72(CCB), T_CCB                     ; 2769
                   52        0C    AB  3C 0005B     MOVZWL  12(CCB), R2                         ; 2770
                   1B              52  B1 0005F     CMPW    R2, #27                             ;
                                    11  12 00062     BNEQ    6$                                  ;
      0C      FE   A0              02  E1 00064     BBC     #2, -2(T_CCB), 6$                   ;
```

```
                        51      22  AB  3C 00069          MOVZWL  34(CCB), R1                  ; 2771
                        51      C8  A0  C0 0006D          ADDL2   -56(T_CCB), R1               ;
                                    51  D6 00071          INCL    R1                           ;
                                    02  11 00073          BRB     7$                           ;
                                    51  D4 00075 6$:      CLRL    R1                           ; 2770
                    C8  A0          51  D0 00077 7$:      MOVL    R1, -56(T_CCB)               ;
            15      FE  AB          04  E1 0007B          BBC     #4, -2(CCB), 9$              ; 2780
                        1B          52  B1 00080          CMPW    R2, #27                      ; 2782
                                    06  12 00083          BNEQ    8$                           ;
                        50      0E  AB  3C 00085          MOVZWL  14(CCB), R0                  ;
                                    0C  11 00089          BRB     10$                          ;
                        0D          52  B1 0008B 8$:      CMPW    R2, #13                      ; 2783
                                    05  12 0008E          BNEQ    9$                           ;
                        50          02  D0 00090          MOVL    #2, R0                       ;
                                    02  11 00093          BRB     10$                          ;
                                    50  D4 00095 9$:      CLRL    R0                           ; 2780
                        51      22  AB  3C 00097 10$:     MOVZWL  34(CCB), R1                  ;
    00000000'   EF      50          51  C1 0009B          ADDL3   R1, R0, RECOUNT              ;
                        50          01  D0 000A3          MOVL    #1, STATUS                   ; 2786
                                    02  11 000A6          BRB     12$                          ; 2716
                                    50  D4 000A8 11$:     CLRL    STATUS                       ; 2789
                                    04  BA 000AA 12$:     POPR    #^M<R2>                      ; 2792
                                        05 000AC          RSB                                  ;

; Routine Size:  173 bytes,    Routine Base:  _BAS$CODE + 03DF

; 1510          2793  1
```

```
; 1512          2794  1 GLOBAL ROUTINE BAS$$REC_RSL9                    ! Read IO_END
; 1513          2795  1     : JSB_REC9 NOVALUE =
; 1514          2796  1
; 1515          2797  1 !++
; 1516          2798  1 ! FUNCTIONAL DESCRIPTION:
; 1517          2799  1 !
; 1518          2800  1 !       BAS$$REC_RSL9 is a no-op!
; 1519          2801  1 !
; 1520          2802  1 ! FORMAL PARAMETERS:
; 1521          2803  1 !
; 1522          2804  1 !       NONE
; 1523          2805  1 !
; 1524          2806  1 ! IMPLICIT INPUTS:
; 1525          2807  1 !
; 1526          2808  1 !       NONE
; 1527          280   1 !
; 1528          2810  1 ! IMPLICIT OUTPUTS:
; 1529          2811  1 !
; 1530          2812  1 ! ROUTINE VALUE:
; 1531          2813  1 !
; 1532          2814  1 !       NONE
; 1533          2815  1 !
; 1534          2816  1 ! SIDE EFFECTS:
; 1535          2817  1 !
; 1536          2818  1 !--
; 1537          2819  1
; 1538          2820  2     BEGIN
; 1539          2821  2     RETURN;
; 1540          2822  1     END;                                        ! End of BAS$$_REC_RSL9


                                        05 00000 BAS$$REC_RSL9::
                                                 RSB                                                    ; 2822

; Routine Size:  1 bytes,    Routine Base:  _BAS$CODE + 048C

; 1541          2823  1
```

```
; 1543          2824  1 GLOBAL ROUTINE BAS$$REC_MIN9                    ! MAT Input IO_END
; 1544          2825  1     : JSB_REC9 NOVALUE =
; 1545          2826  1
; 1546          2827  1 !++
; 1547          2828  1 ! FUNCTIONAL DESCRIPTION:
; 1548          2829  1 !
; 1549          2830  1 !       BAS$$REC_RSL9 is a no-op!
; 1550          2831  1 !
; 1551          2832  1 ! FORMAL PARAMETERS:
; 1552          2833  1 !
; 1553          2834  1 !       NONE
; 1554          2835  1 !
; 1555          2836  1 ! IMPLICIT INPUTS:
; 1556          2837  1 !
; 1557          2838  1 !       NONE
; 1558          2839  1 !
; 1559          2840  1 ! IMPLICIT OUTPUTS:
; 1560          2841  1 !
; 1561          2842  1 ! ROUTINE VALUE:
; 1562          2843  1 !
; 1563          2844  1 !       NONE
; 1564          2845  1 !
; 1565          2846  1 ! SIDE EFFECTS:
; 1566          2847  1 !
; 1567          2848  1 !--
; 1568          2849  1
; 1569          2850  2     BEGIN
; 1570          2851  2     RETURN;
; 1571          2852  1     END;                                        ! End of BAS$$REC_MIN9


                                        05 00000 BAS$$REC_MIN9:.
                                                 RSB                                                    ; 2852

; Routine Size:  1 bytes,    Routine Base:  _BAS$CODE + 048D

; 1572          2853  1
```

```
; 1574          2854  1 GLOBAL ROUTINE BAS$$REC_MLI1                    !  MAT Linput element transmitter
; 1575          2855  1     : JSB_REC1 =
; 1576          2856  1
; 1577          2857  1 !++
; 1578          2858  1 ! FUNCTIONAL DESCRIPTION:
; 1579          2859  1 !
; 1580          2860  1 !       BAS$$REC_MLI1 unconditionally reads one record.  There  s no
; 1581          2861  1 !       continuation character for MAT LINPUT.
; 1582          2862  1 !       Otherwise an error is signalled.
; 1583          2863  1 !       Then return start and end+1 of user
; 1584          2864  1 !       part of record to be processed as input.
; 1585          2865  1 !
; 1586          2866  1 ! FORMAL PARAMETERS:
; 1587          2867  1 !
; 1588          2868  1 !       NONE
; 1589          2869  1 !
; 1590          2870  1 ! IMPLICIT INPUTS:
; 1591          2871  1 !
; 1592          2872  1 !       LUB$W_RBUF_SIZE         Size of record buffer allocated in OPEN.
; 1593          2873  1 !       LUB$A_RBUF_ADR          Address of record buffer from OPEN.
; 1594          2874  1 !       RAB$L_RBF               Pointer to buffer
; 1595          2875  1 !       RAB$W_RSZ               buffer size
; 1596          2876  1 !
; 1597          2877  1 ! IMPLICIT OUTPUTS:
; 1598          2878  1 !
; 1599          2879  1 !       RECOUNT                 Own storage for RECOUNT function.
; 1600          2880  1 !       LUB$A_BUF_PTR           points to first char of user part of
; 1601          2881  1 !                               record buffer.
; 1602          2882  1 !       LUB$A_BUF_END           points to end+1 of user part of
; 1603          2883  1 !                               record buffer.
; 1604          2884  1 !
; 1605          2885  1 ! ROUTINE VALUE:
; 1606          2886  1 !
; 1607          2887  1 !       NONE
; 1608          2888  1 !
; 1609          2889  1 ! SIDE EFFECTS:
; 1610          2890  1 !
; 1611          2891  1 !       Reads next record from file on this logical unit.
; 1612          2892  1 !       SIGNALs any resultant RMS errors.
; 1613          2893  1 !--
; 1614          2894  1
; 1615          2895  2     BEGIN
; 1616          2896  2
; 1617          2897  2     EXTERNAL REGISTER
; 1618          2898  2         CCB : REF BLOCK [, BYTE];
; 1619          2899  2
; 1620          2900  2     LOCAL
; 1621          2901  2         RMS_STATUS,
; 1622          2902  2         T_CCB : REF BLOCK [, BYTE];
; 1623          2903  2
; 1624          2904  2     LITERAL
; 1625          2905  2         K_ESCAPE = %X'1B',
; 1626          2906  2         K_CR = %X'0D';
; 1627          2907  2
; 1628          2908  2     !+
; 1629          2909  2     ! Read record into buffer using RMS and check for errors
; 1630          2910  2     ! Signal any RMS errors directly.
```

```
; 1631          2911  2      !-
; 1632          2912  2
; 1633          2913  2      RMS_STATUS = $GET (RAB = .CCB);
; 1634          2914  2
; 1635          2915  2      IF .RMS_STATUS EQL RMS$_CONTROLC
; 1636          2916  2      THEN
; 1637          2917  2          BAS$$SIGNAL_CTRLC ();
; 1638          2918  2
; 1639          2919  2      IF NOT .RMS_STATUS
; 1640          2920  2      THEN
; 1641          2921  2          GET_ERROR (K_STOP);
; 1642          2922  2
; 1643          2923  2  !+
; 1644          2924  2  ! Return start-1 and end+1 address of record just read
; 1645          2925  2  ! LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
; 1646          2926  2  ! Input.  This is seen as a solution to the problem of the user entering
; 1647          2927  2  ! <return> as the response to a prompt (null input record) and an empty
; 1648          2928  2  ! or depleted buffer which requires another Get.
; 1649          2929  2  ! The algorithm:
; 1650          2930  2  ! 1)     Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1651          2931  2  !        T: The buffer is depleted - another Get is required.
; 1652          2932  2  ! 2)     Add one to LUB$A_BUF_PTR
; 1653          2933  2  ! 3)     Does LUB$A_BUF_PTR = LUB$A_BUF_END?
; 1654          2934  2  !        T: Return the default value.
; 1655          2935  2  ! 4)     Scan for the next field.
; 1656          2936  2  !
; 1657          2937  2  !-
; 1658          2938  2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
; 1659          2939  2      CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
; 1660          2940  2  !+
; 1661          2941  2  ! Update the cursor position if this input was terminated by an escape.
; 1662          2942  2  ! Save the cursor position if last PRINT terminator was a semi or comma.
; 1663          2943  2  ! Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
; 1664          2944  2  !-
; 1665          2945  2      T_CCB = .CCB [LUB$A_BUDDY_PTR];
; 1666          2946  3      T_CCB [LUB$L_PRINT_POS] = (IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
; 1667          2947  3                                 THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
; 1668          2948  2                                 ELSE 0);
; 1669          2949  2  !+
; 1670          2950  2  ! Set RECOUNT to the number of bytes read
; 1671          2951  2  ! If the file is a terminal format file, then RECOUNT has to be
; 1672          2952  2  ! adjusted for the carriage control terminator.
; 1673          2953  2  !-
; 1674          2954  3      RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
; 1675          2955  3              SET
; 1676          2956  3              [K_ESCAPE] : .CCB [RAB$W_STV2];
; 1677          2957  3              [K_CR] : 2;
; 1678          2958  3              [OTHERWISE] : 0;
; 1679          2959  2              TES ELSE 0);
; 1680          2960  2      RETURN 1
; 1681          2961  1      END;                                        ! End of BAS$$REC_MLI1
```

```
                                                  PUSHL   R2                                  ; 2854
                                  5B  DD 00002    PUSHL   CCB                                 ; 2913
          00000000G  00           01  FB 00004    CALLS   #1, SYS$GET
                     52           50  D0 0000B    MOVL    R0, RMS_STATUS
          00010651   8F           52  D1 0000E    CMPL    RMS_STATUS, #67153                  ; 2915
                                  07  12 00015    BNEQ    1$
          00000000G  00           00  FB 00017    CALLS   #0, BAS$$SIGNAL_CTRLC               ; 2917
                     07           52  E8 0001E 1$: BLBS   RMS_STATUS, 2$                      ; 2919
                                  7E  D4 00021    CLRL    -(SP)                               ; 2921
             0000V  CF            01  FB 00023    CALLS   #1, GET_ERROR
B0   AB        28   AB            01  C3 00028 2$: SUBL3  #1, 40(CCB), -80(CCB)               ; 2938
                    50       22  AB  3C 0002E    MOVZWL  34(CCB), R0                         ; 2939
               B4   AB       28 BB40  9E 00032    MOVAB   @40(CCB)[R0], -76(CCB)
                    51       B8  AB  D0 00038    MOVL    -72(CCB), T_CCB                     ; 2945
                    52       0C  AB  3C 0003C    MOVZWL  12(CCB), R2                         ; 2946
                    1B           52  B1 00040    CMPW    R2, #27
                                 11  12 00043    BNEQ    3$
     0C        FE   A1           02  E1 00045    BBC     #2, -2(T_CCB), 3$
                    50       22  AB  3C 0004A    MOVZWL  34(CCB), R0                         ; 2947
                    50       C8  A1  C0 0004E    ADDL2   -56(T_CCB), R0
                                 50  D6 00052    INCL    R0
                                 02  11 00054    BRB     4$
                                 50  D4 00056 3$: CLRL   R0                                  ; 2946
               C8   A1           50  D0 00058 4$: MOVL   R0, -56(T_CCB)
     15        FE   AB           04  E1 0005C    BBC     #4, -2(CCB), 6$                     ; 2954
                    1B           52  B1 00061    CMPW    R2, #27                             ; 2956
                                 06  12 00064    BNEQ    5$
                    50       0E  AB  3C 00066    MOVZWL  14(CCB), R0
                                 0C  11 0006A    BRB     7$
                    0D           52  B1 0006C 5$: CMPW   R2, #13                             ; 2957
                                 05  12 0006F    BNEQ    6$
                    50           02  D0 00071    MOVL    #2, R0
                                 02  11 00074    BRB     7$
                                 50  D4 00076 6$: CLRL   R0                                  ; 2954
                    51       22  AB  3C 00078 7$: MOVZWL  34(CCB), R1
   00000000'   EF   50           51  C1 0007C    ADDL3   R1, R0, RECOUNT
                    50           01  D0 00084    MOVL    #1, R0                              ; 2960
                                 04  BA 00087    POPR    #^M<R2>                             ; 2961
                                     05 00089    RSB                                         ;

; Routine Size:  138 bytes,    Routine Base:  _BAS$CODE + 048E

; 1682          2962  1
```

```
1 GLOBAL ROUTINE BAS$$REC_WSLO                          ! Write list-directed
1     : JSB_REC0 NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       BAS$$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 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
1 !       LUB$V_FIXED              1 if fixed-length records
1 !       LUB$V_FORM_CHAR         Indicates that the last element transmitter ended
1 !                               in a comma or semicolon format char.
1 !       LUB$V_FORCIBLE          Indicates a forcible device
1 !       LUB$V_CCO               Cancel control O
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       LUB$B_BAS_VFC1          'Pre' carriage control
1 !       LUB$B_BAS_VFC2          'Post' carriage control
1 !       LUB$A_BUF_PTR           pointer to next byte of buffer
1 !       LUB$A_BUF_END           pointer to byte following the buffer
1 !       RAB$V_CCO               Cancel control O
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2 !+
2 ! Copy the current status of the cancel-control-o bit in the LUB
2 ! (possibly set by RCTRLO) into the RAB, and clear it from the
2 ! LUB.  The net effect of this is that if the bit is set in the
2 ! LUB, then the CANCTRLO modifier will be applied to this write
2 ! operation only.
2 !-
2
2     CCB [RAB$V_CCO] = .CCB [LUB$V_CCO];
2     CCB [LUB$V_CCO] = 0;
2
2     !+
```

```
; 1741          3020  2     ! If there is a format character pending and this is not a forcible
; 1742          3021  2     ! device, then don't change the buffer pointers.  The PUT will be done when
; 1743          3022  2     ! there is no format character pending.
; 1744          3023  2     !-
; 1745          3024  2
; 1746          3025  2     IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;
; 1747          3026  2
; 1748          3027  2     !+
; 1749          3028  2     ! If the last statement did not end with a format character,
; 1750          3029  2     ! then put a line feed into the 'pre' carriage control
; 1751          3030  2     ! Unconditionally set the 'post' carriage control to null
; 1752          3031  2     !-
; 1753          3032  2
; 1754          3033  2     CCB [LUB$B_BAS_VFC1] = (IF .CCB [LUB$V_FORM_CHAR] THEN BAS$K_NULL ELSE BAS$K_LF);
; 1755          3034  2     CCB [LUB$B_BAS_VFC2] = BAS$K_NULL;
; 1756          3035  2
; 1757          3036  2     !+
; 1758          3037  2     ! If the buffer is dirty, then this is recursive I/O and we want to
; 1759          3038  2     ! concatenate this record.  So leave the buffer pointers alone. Otherwise
; 1760          3039  2     ! return the buffer pointers initialized for another statement
; 1761          3040  2     !-
; 1762          3041  2
; 1763          3042  2     IF NOT .CCB [LUB$V_OUTBUF_DR]
; 1764          3043  2     THEN
; 1765          3044  3         BEGIN
; 1766          3045  3         CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
; 1767          3046  3         CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
; 1768          3047  2         END;
; 1769          3048  2
; 1770          3049  2     RETURN;
; 1771          3050  1     END;                                        ! END OF BAS$$REC_WSLO


        50      A0  AB          01          02  EF 00000 BAS$$REC_WSLO::
                                                        EXTZV   #2, #1, -96(CCB), R0                    ; 3016
    07  AB          01          07          50  F0 00006        INSV    R0, #7, #1, 7(CCB)              ;
                        A0  AB          04  8A 0000C        BICB2   #4, -96(CCB)                        ; 3017
                    0E  FE  AB          02  E1 00010        BBC     #2, -2(CCB), 1$                     ; 3025
                    24  FE  AB          06  E1 00015        BBC     #6, -2(CCB), 3$                     ;
                    04  FE  AB          02  E1 0001A        BBC     #2, -2(CCB), 1$                     ; 3033
                                        50  D4 0001F        CLRL    R0                                  ;
                                        03  11 00021        BRB     2$                                  ;
                            50          01  D0 00023 1$:    MOVL    #1, R0                              ;
                        DA  AB          50  9B 00026 2$:    MOVZBW  R0, -38(CCB)                        ;
                    0F  FE  AB          03  E0 0002A        BBS     #3, -2(CCB), 3$                     ; 3042
                        B0  AB      EC  AB  D0 0002F        MOVL    -20(CCB), -80(CCB)                  ; 3045
                            50      D2  AB  3C 00034        MOVZWL  -46(CCB), R0                        ; 3046
                        B4  AB      EC BB40 9E 00038        MOVAB   @-20(CCB)[R0], -76(CCB)             ;
                                            05 0003E 3$:    RSB                                         ; 3050

; Routine Size:  63 bytes,    Routine Base:  _BAS$CODE + 0518

; 1772          3051  1
```

```
1774    3052  1 GLOBAL ROUTINE BAS$$REC_WSL1 (                    ! Write list-directed
1775    3053  1         CARRIAGE_CTRL) : JSB_REC_WSL1 NOVALUE =          ! Called from BAS$$DO_WRITE
1776    3054  1
1777    3055  1 !++
1778    3056  1 ! FUNCTIONAL DESCRIPTION:
1779    3057  1 !
1780    3058  1 !       Write one list-directed record and initialize for the next
1781    3059  1 !       BAS$$REC_WSL1 writes one output buffer and then
1782    3060  1 !       initializes the output buffer and returns start and end+1 of user
1783    3061  1 !       part of record buffer to be filled by caller.
1784    3062  1 !       If this routine is called because the buffer overflowed then the 'post'
1785    3063  1 !       carriage control should be null.  If this routine is called because the
1786    3064  1 !       margin overflowed, then the 'post' carriage control should be 'CR'.
1787    3065  1 !
1788    3066  1 ! FORMAL PARAMETERS:
1789    3067  1 !
1790    3068  1 !       CARRIAGE_CTRL.rlu.v     carriage control for the record
1791    3069  1 !
1792    3070  1 ! IMPLICIT INPUTS:
1793    3071  1 !
1794    3072  1 !       LUB$W_RBUF_SIZE         Size (bytes) allocated for record buffer at OPEN.
1795    3073  1 !       LUB$A_RBUF_ADR          Address of record buffer allocated at OPEN
1796    3074  1 !       LUB$A_BUF_PTR           Pointer to next byte in user buffer.
1797    3075  1 !       RAB$L_RBF               Pointer to user buffer
1798    3076  1 !
1799    3077  1 ! IMPLICIT OUTPUTS:
1800    3078  1 !
1801    3079  1 !       LUB$B_BAS_VFC1          'Pre' carriage control
1802    3080  1 !       LUB$B_BAS_VFC2          'Post' carriage control
1803    3081  1 !       LUB$A_BUF_PTR           Pointer to start of user part of record buffer
1804    3082  1 !       LUB$A_BUF_END           Pointer to end+1 of user part of record buffer
1805    3083  1 !       LUB$V_OUTBUF_DR         Indicates valid data in the output buffer
1806    3084  1 !       RAB$W_RSZ               Size of user buffer
1807    3085  1 !
1808    3086  1 ! ROUTINE VALUE:
1809    3087  1 !
1810    3088  1 !       NONE
1811    3089  1 !
1812    3090  1 ! SIDE EFFECTS:
1813    3091  1 !
1814    3092  1 !       Writes one RMS sequential record.
1815    3093  1 !       SIGNALs BAS$K_FATSYSIO on PUT error.
1816    3094  1 !--
1817    3095  1
1818    3096  2     BEGIN
1819    3097  2
1820    3098  2     EXTERNAL REGISTER
1821    3099  2         CCB = 11 : REF BLOCK [, BYTE];
1822    3100  2
1823    3101  2     LITERAL
1824    3102  2         K_NO_CR = 2;
1825    3103  2
1826    3104  2     LOCAL
1827    3105  2         RMS_STATUS;
1828    3106  2
1829    3107  2 !+
1830    3108  2 ! Set 'post' carriage control to CR or NULL depending on whether the margin
```

```
; 1831          3109  2 ! overflowed or the buffer overflowed.
; 1832          3110  2 ! If this is a file, the carriage control is ignored and a record is PUT.
; 1833          3111  2 !-
; 1834          3112  2     CCB [LUB$B_BAS_VFC2] = (IF .CARRIAGE_CTRL EQL BAS$K_BUF_EXC THEN BAS$K_NULL ELSE BAS$K_CR);
; 1835          3113  2
; 1836          3114  2     !+
; 1837          3115  2     ! perform the record write.
; 1838          3116  2     ! Set recordsize to actual length of record
; 1839          3117  2     !-
; 1840          3118  2
; 1841          3119  2     CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
; 1842          3120  2
; 1843          3121  2     !+
; 1844          3122  2     ! Output buffer to RMS and check for errors
; 1845          3123  2     ! If errors, SIGNAL_STO
; 1846          3124  2     !-
; 1847          3125  2
; 1848          3126  2     CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];
; 1849          3127  2     CCB [LUB$V_OUTBUF_DR] = 0;
; 1850          3128  2
; 1851          3129  2     RMS_STATUS = $PUT (RAB = .CCB);
; 1852          3130  2
; 1853          3131  2     IF .RMS_STATUS EQL RMS$_CONTROLC
; 1854          3132  2     THEN
; 1855          3133  2         BAS$$SIGNAL_CTRLC ();
; 1856          3134  2
; 1857          3135  2     IF NOT .RMS_STATUS
; 1858          3136  2     THEN
; 1859          3137  2         PUT_ERROR (K_STOP);
; 1860          3138  2
; 1861          3139  2     !+
; 1862          3140  2     ! Set the 'pre' carriage control to LF if CARRIAGE_CTRL warrants it.
; 1863          3141  2     ! Set the 'post' carriage control to null.
; 1864          3142  2     !-
; 1865          3143  2
; 1866          3144  2     CCB [LUB$B_BAS_VFC1] = (IF .CARRIAGE_CTRL EQL BAS$K_BUF_EXC THEN BAS$K_NULL ELSE BAS$K_LF);
; 1867          3145  2     CCB [LUB$B_BAS_VFC2] = BAS$K_NULL;
; 1868          3146  2
; 1869          3147  2     !+
; 1870          3148  2     ! Initialize record buffer for another list-directed write
; 1871          3149  2     ! return record buffer pointers to caller
; 1872          3150  2     !-
; 1873          3151  2
; 1874          3152  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
; 1875          3153  2     CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
; 1876          3154  2     RETURN;
; 1877          3155  1     END;                                        ! End of routine - BAS$$UDF_WSL1


                        0C  BB 00000 BAS$$REC_WSL1::
                                             PUSHR   #^M<R2,R3>                                  ; 3052
                        53  D4 00002         CLRL    R3                                          ; 3112
            08          50  D1 00004         CMPL    CARRIAGE_CTRL, #8                           ;
                        06  12 00007         BNEQ    1$                                          ;
```

```
                                        53  D6 00009          INCL    R3
                                        50  D4 0000B          CLRL    R0
                                        04  11 0000D          BRB     2$
                        50      8D      8F  9A 0000F 1$:      MOVZBL  #141, R0
                    DB  AB              50  90 00013 2$:      MOVB    R0, -37(CCB)
22  AB              B0  AB      BC      AB  A3 00017          SUBW3   -68(CCB), -80(CCB), 34(CCB)          3119
                    28  AB      BC      AB  D0 0001E          MOVL    -68(CCB), 40(CCB)                    3126
                    FE  AB              08  8A 00023          BICB2   #8, -2(CCB)                          3127
                                        5B  DD 00027          PUSHL   CCB                                  3129
            00000000G   00              01  FB 00029          CALLS   #1, SYS$PUT
                        52              50  D0 00030          MOVL    R0, RMS_STATUS
            00010651    8F              52  D1 00033          CMPL    RMS_STATUS, #67153                   3131
                                        07  12 0003A          BNEQ    3$
            00000000G   00              00  FB 0003C          CALLS   #0, BAS$$SIGNAL_CTRLC                3133
                        07              52  E8 00043 3$:      BLBS    RMS_STATUS, 4$                       3135
                                        7E  D4 00046          CLRL    -(SP)                                3137
                0000V   CF              01  FB 00048          CALLS   #1, PUT_ERROR
                        04              53  E9 0004D 4$:      BLBC    R3, 5$                               3144
                                        50  D4 00050          CLRL    R0
                                        03  11 00052          BRB     6$
                        50              01  D0 00054 5$:      MOVL    #1, R0
                    DA  AB              50  9B 00057 6$:      MOVZBW  R0, -38(CCB)
                    B0  AB      EC      AB  D0 0005B          MOVL    -20(CCB), -80(CCB)                   3152
                        50      D2      AB  3C 00060          MOVZWL  -46(CCB), R0                         3153
                    B4  AB      EC  BB40    9E 00064          MOVAB   @-20(CCB)[R0], -76(CCB)
                                        0C  BA 0006A          POPR    #^M<R2,R3>                           3155
                                            05 0006C          RSB

; Routine Size:  109 bytes,    Routine Base:  _BAS$CODE + 0557

; 1878          3156  1
```

```
1 GLOBAL ROUTINE BAS$$REC_RMFO                    ! Initialize read memory formatted
1     : JSB_REC0 NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Pick up pointer to last major frame from ISB and initialize BUF_BEG,
1 !       BUF_PTR, and BUF_END to the values for the data area found in the
1 !       frame.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       ISB$A_MAJ_F_PTR                 pointer to last Basic major frame
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
2
2     LOCAL
2         BMF : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
2
2     !+
2     ! Reach back into the last major frame by picking up the value of R11 stored
2     ! in the ISB.  Initialize BUF_PTR, BUF_END, BUF_BEG so that this will look
2     ! like a vanilla INPUT.
2     !-
2
2     BMF = .CCB [ISB$A_MAJ_F_PTR];
2 !+
2 ! If this cell is zero, then there was no DATA statement and an error should be
2 ! signalled.
2 !-
2
2     IF .BMF [BSF$A_CUR_DTA] EQLA 0 THEN BAS$$STOP_IO (BAS$K_OUTOF_DAT);
2
2     CCB [LUB$A_BUF_BEG] = .BMF [BSF$A_CUR_DTA];
2     CCB [LUB$A_BUF_END] = .BMF [BSF$A_END_DTA];
2
2     !+
2     ! Subtract one from CUR_DATA for INPUT element transmitter compatibility.
```

```
; 1937          3214  2     !-
; 1938          3215  2
; 1939          3216  2     CCB [LUB$A_BUF_PTR] = .BMF [BSF$A_CUR_DTA] - 1;
; 1940          3217  2     RETURN;
; 1941          3218  1     END;


                                        0C  BB 00000 BAS$$REC_RMF0::
                                                     PUSHR   #^M<R2,R3>                          ; 3157
                        52     FF48     CB  D0 00002         MOVL    -184(CCB), BMF              ; 3201
                        53     0087     C2  D0 00007         MOVL    135(BMF), R3                ; 3207
                                        0B  12 0000C         BNEQ    1$                          ;
                        7E       00G    8F  9A 0000E         MOVZBL  #BAS$K_OUTOF_DAT, -(SP)     ;
                00000000G  00           01  FB 00012         CALLS   #1, BAS$$STOP_IO            ;
                      BC   AB           53  D0 00019 1$:     MOVL    R3, -68(CCB)                ; 3209
                      B4   AB     008B  C2  D0 0001D         MOVL    139(BMF), -76(CCB)          ; 3210
                      B0   AB       FF  A3  9E 00023         MOVAB   -1(R3), -80(CCB)            ; 3216
                                        0C  BA 00028         POPR    #^M<R2,R3>                  ; 3218
                                            05 0002A         RSB                                 ;

; Routine Size:  43 bytes,    Routine Base:  _BAS$CODE + 05C4

; 1942          3219  1
```

```
; 1944          3220  1 GLOBAL ROUTINE BAS$$REC_MRE1                    ! Mat Read element transmitter
; 1945          3221  1     : JSB_REC1 =
; 1946          3222  1
; 1947          3223  1 !++
; 1948          3224  1 ! FUNCTIONAL DESCRIPTION:
; 1949          3225  1 !
; 1950          3226  1 !       Since MAT READ just takes as much input data as it can get, it will just
; 1951          3227  1 !       return a failure here because there is no more data.
; 1952          3228  1 !
; 1953          3229  1 ! FORMAL PARAMETERS:
; 1954          3230  1 !
; 1955          3231  1 !       NONE
; 1956          3232  1 !
; 1957          3233  1 ! IMPLICIT INPUTS:
; 1958          3234  1 !
; 1959          3235  1 !       NONE
; 1960          3236  1 !
; 1961          3237  1 ! IMPLICIT OUTPUTS:
; 1962          3238  1 !
; 1963          3239  1 !       NONE
; 1964          3240  1 !
; 1965          3241  1 ! ROUTINE VALUE:
; 1966          3242  1 !
; 1967          3243  1 !       Returns failure - out of data.
; 1968          3244  1 !
; 1969          3245  1 ! SIDE EFFECTS:
; 1970          3246  1 !
; 1971          3247  1 !       As a result of the failure being returned, the MAT READ will stop
; 1972          3248  1 !       filling the matrix.
; 1973          3249  1 !
; 1974          3250  1 !--
; 1975          3251  1
; 1976          3252  2     BEGIN
; 1977          3253  2     RETURN 0
; 1978          3254  1     END;                                        ! end of BAS$$REC_MRE1



                                        50  D4 00000 BAS$$REC_MRE1::
                                                     CLRL    R0                                          ; 3253
                                            05 00002 RSB                                                 ; 3254

; Routine Size:  3 bytes,    Routine Base:  _BAS$CODE + 05EF

; 1979          3255  1
```

```
; 1981          3256  1 GLOBAL ROUTINE BAS$$REC_RMF1                    ! Read element transmitter
; 1982          3257  1     : JSB_REC1 NOVALUE =
; 1983          3258  1
; 1984          3259  1 !++
; 1985          3260  1 ! FUNCTIONAL DESCRIPTION:
; 1986          3261  1 !
; 1987          3262  1 !       BAS$$REC_RMF1 should not be called in normal processing and will signal
; 1988          3263  1 !       an error (BAS$K_OUTOF_DAT) if it is called.
; 1989          3264  1 !
; 1990          3265  1 ! FORMAL PARAMETERS:
; 1991          3266  1 !
; 1992          3267  1 !       NONE
; 1993          3268  1 !
; 1994          3269  1 ! IMPLICIT INPUTS:
; 1995          3270  1 !
; 1996          3271  1 !       NONE
; 1997          3272  1 !
; 1998          3273  1 ! IMPLICIT OUTPUTS:
; 1999          3274  1 !
; 2000          3275  1 ! ROUTINE VALUE:
; 2001          3276  1 !
; 2002          3277  1 !       NONE
; 2003          3278  1 !
; 2004          3279  1 ! SIDE EFFECTS:
; 2005          3280  1 !
; 2006          3281  1 !       Signal - BAS$K_OUTOF_DAT - Out of Data
; 2007          3282  1 !
; 2008          3283  1 !--
; 2009          3284  1
; 2010          3285  2     BEGIN
; 2011          3286  2     BAS$$SIGNAL (BAS$K_OUTOF_DAT);
; 2012          3287  2     RETURN;
; 2013          3288  1     END;                                        ! end of BAS$$REC_RMF1


                                7E        00G  8F  9A 00000 BAS$$REC_RMF1::
                                                            MOVZBL  #BAS$K_OUTOF_DAT, -(SP)                    ; 3286
                        00000000G  00             01  FB 00004  CALLS   #1, BAS$$SIGNAL                        ;
                                                      05 0000B  RSB                                            ; 3288

; Routine Size:  12 bytes,    Routine Base:  _BAS$CODE + 05F2

; 2014          3289  1
```

```
; 2016          3290  1 GLOBAL ROUTINE BAS$$REC_RMF9                    ! Read IO_END
; 2017          3291  1     : JSB_REC9 NOVALUE =
; 2018          3292  1
; 2019          3293  1 !++
; 2020          3294  1 ! FUNCTIONAL DESCRIPTION:
; 2021          3295  1 !
; 2022          3296  1 !       Update the current data pointer in the last Basic major frame
; 2023          3297  1 !
; 2024          3298  1 ! FORMAL PARAMETERS:
; 2025          3299  1 !
; 2026          3300  1 !       NONE
; 2027          3301  1 !
; 2028          3302  1 ! IMPLICIT INPUTS:
; 2029          3303  1 !
; 2030          3304  1 !       NONE
; 2031          3305  1 !
; 2032          3306  1 ! IMPLICIT OUTPUTS:
; 2033          3307  1 !
; 2034          3308  1 ! ROUTINE VALUE:
; 2035          3309  1 !
; 2036          3310  1 !       NONE
; 2037          3311  1 !
; 2038          3312  1 ! SIDE EFFECTS:
; 2039          3313  1 !
; 2040          3314  1 !--
; 2041          3315  1
; 2042          3316  2     BEGIN
; 2043          3317  2
; 2044          3318  2     EXTERNAL REGISTER
; 2045          3319  2         CCB = K_CCB_REG : REF BLOCK [0, BYTE];
; 2046          3320  2
; 2047          3321  2     LOCAL
; 2048          3322  2         BMF : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
; 2049          3323  2
; 2050          3324  2     !+
; 2051          3325  2     ! Set the current data pointer in the frame to BUF_PTR + 1.
; 2052          3326  2     ! The one is added because Input initialize will subtract one from BUF_PTR.
; 2053          3327  2     ! This whole matter is explained in IO_BEG.
; 2054          3328  2     !-
; 2055          3329  2
; 2056          3330  2     BMF = .CCB [ISB$A_MAJ_F_PTR];
; 2057          3331  2     BMF [BSF$A_CUR_DTA] = .CCB [LUB$A_BUF_PTR] + 1;
; 2058          3332  2     RETURN;
; 2059          3333  1     END;                                        ! End of routine BAS$$REC_RMF9


                                50     FF48   CB  D0 00000 BAS$$REC_RMF9::
                                                           MOVL    -184(CCB), BMF                          ; 3330
                0087   C0       B0     AB          01  C1 00005  ADDL3   #1, -80(CCB), 135(BMF)            ; 3331
                                                       05 0000C  RSB                                       ; 3333

; Routine Size:  13 bytes,    Routine Base:  _BAS$CODE + 05FE
```

; 2060 3334 1

```
; 2062          3335  1 GLOBAL ROUTINE BAS$$REC_GSE (                    ! GET (sequential) a record
; 2063          3336  1         FOREIGN_BUFFER,                          !
; 2064          3337  1         LOCK_FLAGS
; 2065          3338  1     ) : JSB_DO_READ NOVALUE =
; 2066          3339  1
; 2067          3340  1 !++
; 2068          3341  1 ! FUNCTIONAL DESCRIPTION:
; 2069          3342  1 !
; 2070          3343  1 !       Read one record.  Update RECOUNT if successful.
; 2071          3344  1 !       If a foreign buffer is specified, then change RAB$L_RBF to point to the
; 2072          3345  1 !       "foreign buffer".  Otherwise, signal a fatal error.
; 2073          3346  1 !
; 2074          3347  1 ! FORMAL PARAMETERS:
; 2075          3348  1 !
; 2076          3349  1 !       FOREIGN_BUFFER.rl.v             points to CB of foreign buffer or 0
; 2077          3350  1 !       LOCK_FLAGS.rlu.v                bits to set in RAB ROP for manual
; 2078          3351  1 !                                       record locking (0 if none)
; 2079          3352  1 ! IMPLICIT INPUTS:
; 2080          3353  1 !
; 2081          3354  1 !       RAB$W_USZ               User buffer size of foreign buffer
; 2082          3355  1 !       RAB$L_UBF               Pointer to user buffer for foreign buffer
; 2083          3356  1 !       LUB$L_WAIT_TIME         Wait time for next input
; 2084          3357  1 !       WAIT                    The module level OWN WAIT
; 2085          3358  1 !
; 2086          3359  1 ! IMPLICIT OUTPUTS:
; 2087          3360  1 !
; 2088          3361  1 !       RAB$B_RAC               Record access field
; 2089          3362  1 !       RECOUNT                 Own storage for RECOUNT function.
; 2090          3363  1 !       RAB$L_RBF               Record pointer in RAB.
; 2091          3364  1 !       RAB$W_RSZ               Number of bytes read (stored in RECOUNT)
; 2092          3365  1 !
; 2093          3366  1 ! ROUTINE VALUE:
; 2094          3367  1 !
; 2095          3368  1 !       NONE
; 2096          3369  1 !
; 2097          3370  1 ! SIDE EFFECTS:
; 2098          3371  1 !
; 2099          3372  1 !       RAB$W_USZ and RAB$W_UBF are altered while this routine is running,
; 2100          3373  1 !       but are restored before exit.
; 2101          3374  1 !       Reads next record from file on this logical unit.
; 2102          3375  1 !       SIGNALs any RMS errors
; 2103          3376  1 !--
; 2104          3377  1
; 2105          3378  2     BEGIN
; 2106          3379  2
; 2107          3380  2     EXTERNAL REGISTER
; 2108          3381  2         CCB : REF BLOCK [, BYTE];
; 2109          3382  2
; 2110          3383  2     MAP
; 2111          3384  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2112          3385  2
; 2113          3386  2     LOCAL
; 2114          3387  2         RMS_STATUS,
; 2115          3388  2         SAVE_USZ,                           ! Save the USZ
; 2116          3389  2         ACTUAL_RSZ,                         ! Actual record size
; 2117          3390  2         WAIT_TIME;                          ! Current wait time
; 2118          3391  2 !+
```

```
; 2119          3392  2 ! Save USZ in case it is modified.  It is faster to always
; 2120          3393  2 ! save and restore it, since that eliminates the test for foreign
; 2121          3394  2 ! buffers and single-character mode at the end of this routine.
; 2122          3395  2 !-
; 2123          3396  2     SAVE_USZ = .CCB [RAB$W_USZ];
; 2124          3397  2 !+
; 2125          3398  2 ! If a timeout has been specified, store information in the RAB to tell
; 2126          3399  2 ! RMS about it.  If no timeout has been specified, clear the TMO bit
; 2127          3400  2 ! in case there was an earlier timeout specified.
; 2128          3401  2 !-
; 2129          3402  2
; 2130          3403  2 !+
; 2131          3404  2 ! If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
; 2132          3405  2 ! , i.e. existing EXE's can run with the LUB wait value in V2.2.
; 2133          3406  2 !-
; 2134          3407  2     WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
; 2135          3408  2
; 2136          3409  3     IF (.WAIT_TIME EQL 0)
; 2137          3410  2     THEN
; 2138          3411  2         CCB [RAB$V_TMO] = 0
; 2139          3412  2     ELSE
; 2140          3413  3         BEGIN
; 2141          3414  3         CCB [RAB$B_TMO] = .WAIT_TIME;
; 2142          3415  3         CCB [RAB$V_TMO] = 1;
; 2143          3416  2         END;
; 2144          3417  2
; 2145          3418  2
; 2146          3419  2 !+
; 2147          3420  2 ! Set the Read-no-echo RMS bit based on the user's last call to
; 2148          3421  2 ! ECHO or NOECHO.
; 2149          3422  2 !-
; 2150          3423  2     CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
; 2151          3424  2 !+
; 2152          3425  2 ! Set the record pointer field in the RAB to the user buffer.  This is
; 2153          3426  2 ! done on each element transmission and not just at OPEN because of RMS
; 2154          3427  2 ! Locate mode.
; 2155          3428  2 ! Fill the input buffer with Nulls.  Basic semantics require null fill if
; 2156          3429  2 ! the record read is shorter than the buffer.
; 2157          3430  2 ! Set the record access field in the RAB to sequential.  Perform the GET.
; 2158          3431  2 ! If RMS returns a failure status, signal the error.  If the GET is
; 2159          3432  2 ! successful, then update recount.
; 2160          3433  2 !-
; 2161          3434  2
; 2162          3435  3     IF (.FOREIGN_BUFFER NEQA 0)
; 2163          3436  2     THEN
; 2164          3437  3         BEGIN
; 2165          3438  3 !+
; 2166          3439  3 ! A foreign buffer was specified.  Save off RAB$L_usz of the "file" channel
; 2167          3440  3 ! and then substitute the appropriate values from the foreign channel into
; 2168          3441  3 ! the file channel to make the record go directly into the foreign buffer.
; 2169          3442  3 !-
; 2170          3443  3         CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
; 2171          3444  2         END;
; 2172          3445  2
; 2173          3446  2 !+
; 2174          3447  2 ! If the user has called ONECHR, shrink the effective size of the
; 2175          3448  2 ! buffer to one character, so he will get characters one at a time.
```

```
; 2176          3449  2 ! The user must call ONECHR before each GET, so we clear the ONECHR
; 2177          3450  2 ! flag here.
; 2178          3451  2 !-
; 2179          3452  2
; 2180          3453  3     IF (.CCB [LUB$V_ONECHR])
; 2181          3454  2     THEN
; 2182          3455  3         BEGIN
; 2183          3456  3         CCB [LUB$V_ONECHR] = 0;
; 2184          3457  3         CCB [RAB$W_USZ] = 1;
; 2185          3458  2         END;
; 2186          3459  2
; 2187          3460  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 2188          3461  2
; 2189          3462  2 !+
; 2190          3463  2 ! Set bits in the RAB ROP (careful not to turn off ULK).
; 2191          3464  2 !-
; 2192          3465  2
; 2193          3466  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 2194          3467  2
; 2195          3468  2     RMS_STATUS = $GET (RAB = .CCB);
; 2196          3469  2
; 2197          3470  2     IF .RMS_STATUS EQL RMS$_CONTROLC
; 2198          3471  2     THEN
; 2199          3472  2         BAS$$SIGNAL_CTRLC ();
; 2200          3473  2
; 2201          3474  2     IF NOT .RMS_STATUS
; 2202          3475  2     THEN
; 2203          3476  3         BEGIN
; 2204          3477  3 !+
; 2205          3478  3 ! We cannot call GET_ERROR because we must restore UBF and USZ.
; 2206          3479  3 !-
; 2207          3480  3
; 2208          3481  3         WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2209          3482  4             BEGIN
; 2210          3483  4             $WAIT (RAB = .CCB);
; 2211          3484  4             $GET (RAB = .CCB);
; 2212          3485  3             END;
; 2213          3486  3
; 2214          3487  2         END;
; 2215          3488  2
; 2216          3489  2 !+
; 2217          3490  2 ! Clear RAB ROP bits so that a subsequent I/O operation does not
; 2218          3491  2 ! inherit them.
; 2219          3492  2 !-
; 2220          3493  2
; 2221          3494  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 2222          3495  2
; 2223          3496  2 !+
; 2224          3497  2 ! This actual record size may or may not change below.  If the file is a
; 2225          3498  2 ! terminal device then it will get terminators tacked on to the record read.
; 2226          3499  2 !-
; 2227          3500  2     ACTUAL_RSZ = .CCB [RAB$W_RSZ];
; 2228          3501  2
; 2229          3502  2
; 2230          3503  2 !+
; 2231          3504  2 ! Tack on the terminators when a terminal device file, just like INPUT LINE
; 2232          3505  2 !-
```

```
; 2233          3506  2     IF .CCB[LUB$V_TERM_DEV]
; 2234          3507  2     THEN
; 2235          3508  3         BEGIN
; 2236          3509  3         LITERAL K_ESCAPE = %X'1B',
; 2237          3510  3                 K_CR     = %X'0D',
; 2238          3511  3                 K_CRLF   = %X'0A0D';
; 2239          3512  3
; 2240          3513  3 !+
; 2241          3514  3 ! STV0 is the escape character, STV2 is the number of terminating characters.
; 2242          3515  3 !-
; 2243          3516  3         SELECTONEU .CCB [RAB$W_STV0] OF
; 2244          3517  3          SET
; 2245          3518  3          [K_ESCAPE] :
; 2246          3519  4                 BEGIN
; 2247          3520  4 !+
; 2248          3521  4 ! If the length is one then escape is not at the end of the buffer and it
; 2249          3522  4 ! must be moved there, otherwise the escape sequence is at the end of the
; 2250          3523  4 ! buffer following the data.
; 2251          3524  4
; 2252          3525  4                 IF .CCB [RAB$W_STV2] EQLU 1
; 2253          3526  4                 THEN
; 2254          3527  5                    BEGIN
; 2255          3528  5                    IF .CCB [RAB$W_RSZ] GEQU .CCB [RAB$W_USZ]
; 2256          3529  5                    THEN BAS$$STOP_IO (BAS$K_RECFILTOO);
; 2257          3530  5                    CH$MOVE (1,UPLIT(K_ESCAPE),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
; 2258          3531  5                    ACTUAL_RSZ = .ACTUAL_RSZ + 1;
; 2259          3532  5                    END
; 2260          3533  4                 ELSE
; 2261          3534  4                    ACTUAL_RSZ = .ACTUAL_RSZ + .CCB [RAB$W_STV2];
; 2262          3535  3                 END;
; 2263          3536  3          [K_CR] :
; 2264          3537  4                 BEGIN
; 2265          3538  4                 IF .CCB [RAB$W_RSZ] + 2 GTRU .CCB [RAB$W_USZ]
; 2266          3539  4                 THEN BAS$$STOP_IO (BAS$K_RECFILTOO);
; 2267          3540  4                 CH$MOVE (2,UPLIT(K_CRLF),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
; 2268          3541  4                 ACTUAL_RSZ = .ACTUAL_RSZ + 2 ;
; 2269          3542  3                 END;
; 2270          3543  3          [OTHERWISE] :
; 2271          3544  3                 ;
; 2272          3545  3          TES;
; 2273          3546  2         END;
; 2274          3547  2
; 2275          3548  2 !+
; 2276          3549  2 ! If there are no errors, null pad the buffer.
; 2277          3550  2 !-
; 2278          3551  2
; 2279          3552  2     IF (.CCB [RAB$W_USZ] GTR .ACTUAL_RSZ)AND .CCB [RAB$L_STS]
; 2280          3553  2     THEN
; 2281          3554  2         CH$FILL (%X'00',
; 2282          3555  2             .CCB [RAB$W_USZ] - .ACTUAL_RSZ, .CCB [RAB$L_UBF] + .ACTUAL_RSZ);
; 2283          3556  2
; 2284          3557  2 !+
; 2285          3558  2 ! Before checking for errors, restore UBF and USZ, and set RECOUNT.
; 2286          3559  2 !-
; 2287          3560  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 2288          3561  2     CCB [RAB$W_USZ] = .SAVE_USZ;
; 2289          3562  2     RECOUNT = .ACTUAL_RSZ;
```

```
; 2290   3563  2 !+
; 2291   3564  2 ! Any error remaining (which will be an error other than Record Stream
; 2292   3565  2 ! Active, RSA) is fatal.
; 2293   3566  2 !-
; 2294   3567  2
; 2295   3568  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 2296   3569  2
; 2297   3570  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
; 2298   3571  2     RETURN;
; 2299   3572  1     END;                                          ! End of BAS$$REC_GSE


                                        0060B          .BLKB   1
                                 0000001B  0060C P.AAA:  .LONG   27                                  ;
                                 00000A0D  00610 P.AAB:  .LONG   2573                                ;

                                                       .EXTRN  SYS$WAIT

                                    3C  BB 00000 BAS$$REC_GSE::
                                                       PUSHR   #^M<R2,R3,R4,R5>                    ; 3335
                 5E                 10  C2 00002       SUBL2   #16, SP                             ;
                 54                 51  D0 00005       MOVL    R1, R4                              ;
        08   AE        20           AB  9E 00008       MOVAB   32(CCB), 8(SP)                      ; 3396
        04   AE        08           BE  3C 0000D       MOVZWL  @8(SP), SAVE_USZ                    ;
             51 00000000'           EF  3C 00012       MOVZWL  WAIT, R1                            ; 3407
                                    04  12 00019       BNEQ    1$                                  ;
             51        CC           AB  D0 0001B       MOVL    -52(CCB), WAIT_TIME                 ;
             52        04           AB  9F 0001F 1$:   MOVAB   4(CCB), R2                          ; 3411
                                    51  D5 00023       TSTL    WAIT_TIME                           ; 3409
                                    06  12 00025       BNEQ    2$                                  ;
        03   A2                     02  8A 00027       BICB2   #2, 3(R2)                           ; 3411
                                    08  11 0002B       BRB     3$                                  ;
        1F   AB                     51  90 0002D 2$:   MOVB    WAIT_TIME, 31(CCB)                  ; 3414
        03   A2                     02  88 00031       BISB2   #2, 3(R2)                           ; 3415
03   A2        01        00     A0   AB  F0 00035 3$:   INSV    -96(CCB), #0, #1, 3(R2)             ; 3423
                                    50  D5 0003C       TSTL    FOREIGN_BUFFER                      ; 3435
                                    05  13 0003E       BEQL    4$                                  ;
        08   BE        20     A0   B0 00040           MOVW    32(FOREIGN_BUFFER), @8(SP)          ; 3443
        08   A0        AB          01  E1 00045 4$:   BBC     #1, -96(CCB), 5$                    ; 3453
             A0        AB          02  8A 0004A       BICB2   #2, -96(CCB)                        ; 3456
             08        BE          01  B0 0004E       MOVW    #1, @8(SP)                          ; 3457
                                 1E AB  94 00052 5$:   CLRB    30(CCB)                             ; 3460
                       62            54  C8 00055       BISL2   LOCK_FLAGS, (R2)                    ; 3466
                                    5B  DD 00058       PUSHL   CCB                                 ; 3468
          00000000G    00           01  FB 0005A       CALLS   #1, SYS$GET                         ;
                       53           50  D0 00061       MOVL    R0, RMS_STATUS                      ;
          00010651     8F           53  D1 00064       CMPL    RMS_STATUS, #67153                  ; 3470
                                    07  12 0006B       BNEQ    6$                                  ;
          00000000G    00           00  FB 0006D       CALLS   #0, BAS$$SIGNAL_CTRLC               ; 3472
                       1E           53  E8 00074 6$:   BLBS    RMS_STATUS, 8$                      ; 3474
          000182DA     8F     08   AB  D1 00077 7$:   CMPL    8(CCB), #99034                      ; 3481
                                    14  12 0007F       BNEQ    8$                                  ;
                                    5B  DD 00081       PUSHL   CCB                                 ; 3483
          00000000G    00           01  FB 00083       CALLS   #1, SYS$WAIT                        ;
                                    5B  DD 0008A       PUSHL   CCB                                 ; 3484
          00000000G    00           01  FB 0008C       CALLS   #1, SYS$GET                         ;
```

```
                                        E2  11 00093          BRB     7$                                      ; 3481
                                 62     54  CC 00095 8$:      XORL2   LOCK_FLAGS, (R2)                        ; 3494
                                 52  22  AB  3C 00098         MOVZWL  34(CCB), R2                             ; 3500
                              0C AE     52  D0 0009C          MOVL    R2, ACTUAL_RSZ                          ;
                        60    FE AB     05  E1 000A0          BBC     #5, -2(CCB), 13$                        ; 3506
                                 50  0C  AB  3C 000A5         MOVZWL  12(CCB), R0                             ; 3516
                                 1B      50  B1 000A9          CMPW    R0, #27                                 ; 3518
                                         2D  12 000AC          BNEQ    11$                                     ;
                                 01  0E  AB  B1 000AE         CMPW    14(CCB), #1                             ; 3525
                                         1D  12 000B2          BNEQ    10$                                     ;
                                 52  08  BE  B1 000B4         CMPW    @8(SP), R2                              ; 3528
                                         0B  1A 000B8          BGTRU   9$                                      ;
                                 7E  00G 8F  9A 000BA         MOVZBL  #BAS$K_RECFILTOO, -(SP)                 ; 3529
                        00000000G 00     01  FB 000BE         CALLS   #1, BAS$$STOP_IO                        ;
                              24 BB42 FF2F CF  90 000C5 9$:   MOVB    P.AAA, @36(CCB)[R2]                     ; 3530
                                 0C      AE  D6 000CC         INCL    ACTUAL_RSZ                              ; 3531
                                         34  11 000CF          BRB     13$                                     ; 3525
                                 50  0E  AB  3C 000D1 10$:    MOVZWL  14(CCB), R0                             ; 3534
                              0C AE     50  C0 000D5          ADDL2   R0, ACTUAL_RSZ                          ;
                                         2A  11 000D9          BRB     13$                                     ; 3516
                                 0D      50  B1 000DB 11$:    CMPW    R0, #13                                 ; 3536
                                         25  12 000DE          BNEQ    13$                                     ;
                                 50  02  A2  9E 000E0         MOVAB   2(R2), R0                               ; 3538
                 50     08  BE  10      00  ED 000E4          CMPZV   #0, #16, @8(SP), R0                     ;
                                         0B  1E 000EA          BGEQU   12$                                     ;
                                 7E  00G 8F  9A 000EC         MOVZBL  #BAS$K_RECFILTOO, -(SP)                 ; 3539
                        00000000G 00     01  FB 000F0         CALLS   #1, BAS$$STOP_IO                        ;
                        50       52  24  AB  C1 000F7 12$:    ADDL3   36(CCB), R2, R0                         ; 3540
                                 60  FEFC CF  B0 000FC         MOVW    P.AAB, (R0)                             ;
                              0C AE     02  C0 00101          ADDL2   #2, ACTUAL_RSZ                          ; 3541
              0C AE     08  BE  10      00  ED 00105 13$:    CMPZV   #0, #16, @8(SP), ACTUAL_RSZ             ; 3552
                                         19  15 0010C          BLEQ    14$                                     ;
                                 15  08  AB  E9 0010E         BLBC    8(CCB), 14$                             ;
                                 50  08  BE  3C 00112         MOVZWL  @8(SP), R0                              ; 3555
                                 50  0C  AE  C2 00116         SUBL2   ACTUAL_RSZ, R0                          ;
                        6E       24  0C  AE  C1 0011A         ADDL3   ACTUAL_RSZ, 36(R11), (SP)               ;
                 50     00          6E      00  2C 00120         MOVC5   #0, (SP), #0, R0, @0(SP)                ;
                                         00  BE     00125                                                      ;
                              24  AB  9C  AB  D0 00127 14$:    MOVL    -100(CCB), 36(CCB)                      ; 3560
                              08  BE  04  AE  B0 0012C         MOVW    SAVE_USZ, @8(SP)                        ; 3561
                        00000000'  EF  0C  AE  D0 00131         MOVL    ACTUAL_RSZ, RECOUNT                     ; 3562
                                 0A  08  AB  E8 00139         BLBS    8(CCB), 15$                             ; 3568
                                 7E      01  CE 0013D          MNEGL   #1, -(SP)                               ;
                        00000000G 00     01  FB 00140         CALLS   #1, BAS$$STOP_IO                        ;
                              EC  AB  28  AB  D0 00147 15$:    MOVL    40(CCB), -20(CCB)                       ; 3570
                                 5E      10  C0 0014C          ADDL2   #16, SP                                 ; 3572
                                         3C  BA 0014F          POPR    #^M<R2,R3,R4,R5>                        ;
                                             05 00151          RSB                                             ;

; Routine Size:  338 bytes,    Routine Base:  _BAS$CODE + 0614

; 2300          3573  1
```

```
: 2302          3574  1 GLOBAL ROUTINE BAS$$REC_GIN (                    ! GET (indexed) a record
: 2303          3575  1         KEY_NO, REL_OP, KEY, FOREIGN_BUFFER, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
: 2304          3576  1
: 2305          3577  1 !++
: 2306          3578  1 ! FUNCTIONAL DESCRIPTION:
: 2307          3579  1 !
: 2308          3580  1 !       Read one record.  Update RECOUNT if successful.
: 2309          3581  1 !       If a foreign buffer is specified, then change RAB$L_RBF to point to the
: 2310          3582  1 !       "foreign buffer".  Otherwise, signal a fatal error.
: 2311          3583  1 !
: 2312          3584  1 ! FORMAL PARAMETERS:
: 2313          3585  1 !
: 2314          3586  1 !       KEY_NO.rl.v             key of reference number
: 2315          3587  1 !       REL_OP.rl.v             relative relationship of the key
: 2316          3588  1 !       KEY.rt.dx               key to search for
: 2317          3589  1 !       FOREIGN_BUFFER.rl.v             points to CB of foreign buffer or 0
: 2318          3590  1 !       LOCK_FLAGS.rlu.v        bits to set in RAB ROP to control manual record
: 2319          3591  1 !                               locking (0 if none)
: 2320          3592  1 !
: 2321          3593  1 ! IMP_ICIT INPUTS:
: 2322          3594  1 !
: 2323          3595  1 !       RAB$W_USZ               User buffer size of foreign buffer
: 2324          3596  1 !       RAB$L_UBF               Pointer to user buffer for foreign buffer
: 2325          3597  1 !
: 2326          3598  1 ! IMPLICIT OUTPUTS:
: 2327          3599  1 !
: 2328          3600  1 !       RAB$B_RAC               Record access field
: 2329          3601  1 !       RECOUNT                 Own storage for RECOUNT function.
: 2330          3602  1 !       RAB$L_RBF               Record pointer in RAB.
: 2331          3603  1 !       RAB$W_USZ               User buffer size for "file" buffer
: 2332          3604  1 !       RAB$L_UBF               Pointer to user buffer for "file" buffer
: 2333          3605  1 !
: 2334          3606  1 ! ROUTINE VALUE:
: 2335          3607  1 !
: 2336          3608  1 !       NONE
: 2337          3609  1 !
: 2338          3610  1 ! SIDE EFFECTS:
: 2339          3611  1 !
: 2340          3612  1 !       Reads next record from file on this logical unit.
: 2341          3613  1 !       SIGNALs any RMS errors
: 2342          3614  1 !--
: 2343          3615  1
: 2344          3616  2     BEGIN
: 2345          3617  2
: 2346          3618  2     EXTERNAL REGISTER
: 2347          3619  2         CCB : REF BLOCK [, BYTE];
: 2348          3620  2
: 2349          3621  2     MAP
: 2350          3622  2         KEY : REF BLOCK [8, BYTE],              ! descriptor of the key
: 2351          3623  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
: 2352          3624  2
: 2353          3625  2     LITERAL
: 2354          3626  2         K_EQUAL = 0,                            ! keys should be equal
: 2355          3627  2         K_GREATER_EQUAL = 1,                    ! key should be greater than or equal
: 2356          3628  2         K_GREATER_THAN = 2;                     ! key should be greater than
: 2357          3629  2
: 2358          3630  2     LOCAL
```

```
; 2359          3631  2          RMS_STATUS,
; 2360          3632  2          SAVE_USZ;                                  ! Save the USZ
; 2361          3633  2
; 2362          3634  2  !+
; 2363          3635  2  ! Save USZ in case it is modified.  It is faster to always
; 2364          3636  2  ! save and restore it, since that eliminates the test for foreign
; 2365          3637  2  ! buffers and single-character mode at the end of this routine.
; 2366          3638  2  !-
; 2367          3639  2      SAVE_USZ = .CCB [RAB$W_USZ];
; 2368          3640  2
; 2369          3641  2      !+
; 2370          3642  2      ! Set the record pointer field in the RAB to the user buffer.  This is
; 2371          3643  2      ! done on each element transmission and not just at OPEN because of RMS
; 2372          3644  2      ! Locate mode.
; 2373          3645  2      ! Fill the input buffer with Nulls.  Basic semantics require null fill if
; 2374          3646  2      ! the record read is shorter than the buffer.
; 2375          3647  2      ! Set the record access field in the RAB to sequential.  Perform the GET.
; 2376          3648  2      ! If RMS returns a failure status, signal the error.  If the GET is
; 2377          3649  2      ! successful, then update recount.
; 2378          3650  2      !-
; 2379          3651  2
; 2380          3652  2      IF .FOREIGN_BUFFER NEQ 0
; 2381          3653  2      THEN
; 2382          3654  3          BEGIN
; 2383          3655  3  !+
; 2384          3656  3  ! A foreign buffer was specified.  Substitute the appropriate values from the foreign channel into
; 2385          3657  3  ! the file channel to make the record go directly into the foreign buffer.
; 2386          3658  3  !-
; 2387          3659  3          CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
; 2388          3660  3          CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
; 2389          3661  2          END;
; 2390          3662  2
; 2391          3663  2      !+
; 2392          3664  2      ! Set the record access field to key.  Set KBF to the key.  Set KSZ to the
; 2393          3665  2      ! the size of the key passed.  Set the key of reference to the desired key.
; 2394          3666  2      ! Use a case statement to toggle KGT and KGE in the ROP.
; 2395          3667  2      !-
; 2396          3668  2
; 2397          3669  2      CCB [RAB$B_RAC] = RAB$C_KEY;
; 2398          3670  2      CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];
; 2399          3671  2      CCB [RAB$B_KRF] = .KEY_NO;
; 2400          3672  3      CCB [RAB$B_KSZ] = (IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
; 2401          3673  3                         THEN
; 2402          3674  3                              .KEY [DSC$W_LENGTH]
; 2403          3675  3                         ELSE
; 2404          3676  2                              (.KEY [DSC$W_LENGTH]/2 + 1));
; 2405          3677  2
; 2406          3678  2      CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF
; 2407          3679  2          SET
; 2408          3680  2
; 2409          3681  2          [K_EQUAL] :
; 2410          3682  2              CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;
; 2411          3683  2
; 2412          3684  2          [K_GREATER_EQUAL] :
; 2413          3685  3              BEGIN
; 2414          3686  3              CCB [RAB$V_KGE] = 1;
; 2415          3687  3              CCB [RAB$V_KGT] = 0;
```

```
; 2416          3688  2             END;
; 2417          3689  2
; 2418          3690  2         [K_GREATER_THAN] :
; 2419          3691  3             BEGIN
; 2420          3692  3             CCB [RAB$V_KGT] = 1;
; 2421          3693  3             CCB [RAB$V_KGE] = 0;
; 2422          3694  2             END;
; 2423          3695  2         TES;
; 2424          3696  2
; 2425          3697  2 !+
; 2426          3698  2 ! Set bits in RAB ROP without turning off ULK.
; 2427          3699  2 !-
; 2428          3700  2
; 2429          3701  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 2430          3702  2
; 2431          3703  2     RMS_STATUS = $GET (RAB = .CCB);
; 2432          3704  2
; 2433          3705  2     IF .RMS_STATUS EQL RMS$_CONTROLC
; 2434          3706  2     THEN
; 2435          3707  2         BAS$$SIGNAL_CTRLC ();
; 2436          3708  2
; 2437          3709  2     IF NOT .RMS_STATUS
; 2438          3710  2     THEN
; 2439          3711  3         BEGIN
; 2440          3712  3 !+
; 2441          3713  3 ! We cannot call GET_ERROR because we must restore UBF and USZ.
; 2442          3714  3 !-
; 2443          3715  3
; 2444          3716  3         WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2445          3717  4             BEGIN
; 2446          3718  4             $WAIT (RAB = .CCB);
; 2447          3719  4             $GET (RAB = .CCB);
; 2448          3720  3             END;
; 2449          3721  3
; 2450          3722  2         END;
; 2451          3723  2
; 2452          3724  2 !+
; 2453          3725  2 ! Turn off bits in RAB ROP so that subsequent I/O operations can not
; 2454          3726  2 ! inherit them.
; 2455          3727  2 !-
; 2456          3728  2
; 2457          3729  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 2458          3730  2
; 2459          3731  2 !+
; 2460          3732  2 ! If there are no errors, null pad the buffer.
; 2461          3733  2 !-
; 2462          3734  2
; 2463          3735  2     IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
; 2464          3736  2     THEN
; 2465          3737  2         CH$FILL (%X'00',
; 2466          3738  2             .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
; 2467          3739  2
; 2468          3740  2 !+
; 2469          3741  2 ! Before checking for errors, restore UBF and USZ, and set RECOUNT.
; 2470          3742  2 !-
; 2471          3743  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 2472          3744  2     CCB [RAB$W_USZ] = .SAVE_USZ;
```

```
; 2473          3745  2     RECOUNT = .CCB [RAB$W_RSZ];
; 2474          3746  2 !+
; 2475          3747  2 ! Any error remaining (which will be an error other than Record Stream
; 2476          3748  2 ! Active, RSA) is fatal.
; 2477          3749  2 !-
; 2478          3750  2
; 2479          3751  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 2480          3752  2
; 2481          3753  2 !+
; 2482          3754  2 ! This is frosting on the cake.  LUB$A_RBUF_ADR points to the record buffer for
; 2483          3755  2 ! MOVE.  The buffer may change due to RMS Locate Mode.  Currently, Locate Mode
; 2484          3756  2 ! is not performed on files which UPDATE or PUT.  However, in anticipation that
; 2485          3757  2 ! RMS may add such a capability, we point RBUF_ADR off to the buffer used by PUT.
; 2486          3758  2 !-
; 2487          3759  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
; 2488          3760  2     RETURN;
; 2489          3761  1     END;                                        ! End of BAS$$REC_GIN


                                    3C  BB 00000 BAS$$REC_GIN::
                                                 PUSHR   #^M<R2,R3,R4,R5>                          ; 3574
                 7E        20       AB  3C 00002         MOVZWL  32(CCB), SAVE_USZ                  ; 3639
                                    53  D5 00006         TSTL    FOREIGN_BUFFER                     ; 3652
                                    0A  13 00008         BEQL    1$                                 ;
            24   AB        24       A3  D0 0000A         MOVL    36(FOREIGN_BUFFER), 36(CCB)        ; 3659
            20   AB        20       A3  B0 0000F         MOVW    32(FOREIGN_BUFFER), 32(CCB)        ; 3660
            1E   AB                 01  90 00014 1$:     MOVB    #1, 30(CCB)                        ; 3669
            30   AB        04       A2  D0 00018         MOVL    4(KEY), 48(CCB)                    ; 3670
            35   AB                 50  90 0001D         MOVB    KEY_NO, 53(CCB)                    ; 3671
                 15        02       A2  91 00021         CMPB    2(KEY), #21                        ; 3672
                                    05  13 00025         BEQL    2$                                 ;
                 52                 62  3C 00027         MOVZWL  (KEY), R2                          ; 3674
                                    08  11 0002A         BRB     3$                                 ;
                 52                 62  3C 0002C 2$:     MOVZWL  (KEY), R2                          ; 3676
                 52                 02  C6 0002F         DIVL2   #2, R2                             ;
                                    52  D6 00032         INCL    R2                                 ;
            34   AB                 52  90 00034 3$:     MOVB    R2, 52(CCB)                        ; 3672
                 52        04       AB  9E 00038         MOVAB   4(CCB), R2                         ; 3682
      02         00                 51  CF 0003C         CASEL   REL_OP, #0, #2                     ; 3678
    0018       000D             0006     00040 4$:        .WORD   5$-4$,-                            ;
                                                                   6$-4$,-                            ;
                                                                   7$-4$                              ;
            02   A2        40       8F  8A 00046 5$:     BICB2   #64, 2(R2)                         ; 3682
                                    10  11 0004B         BRB     8$                                 ;
            02   A2                 20  88 0004D 6$:     BISB2   #32, 2(R2)                         ; 3686
            02   A2        40       8F  8A 00051         BICB2   #64, 2(R2)                         ; 3687
                                    09  11 00056         BRB     9$                                 ; 3678
            02   A2        40       8F  88 00058 7$:     BISB2   #64, 2(R2)                         ; 3692
            02   A2                 20  8A 0005D 8$:     BICB2   #32, 2(R2)                         ; 3693
                 62                 54  C8 00061 9$:     BISL2   LOCK_FLAGS, (R2)                   ; 3701
                                    5B  DD 00064         PUSHL   CCB                                ; 3703
     000000006   00                 01  FB 00066         CALLS   #1, SYS$GET                        ;
                 53                 50  D0 0006D         MOVL    R0, RMS_STATUS                     ;
     00010651    8F                 53  D1 00070         CMPL    RMS_STATUS, #67153                 ; 3705
```

```
                                    07  12 00077          BNEQ    10$                                  ;
                00000000G  00       00  FB 00079          CALLS   #0, BAS$$SIGNAL_CTRLC                ; 3707
                           1E       53  E8 00080 10$:     BLBS    RMS_STATUS, 12$                      ; 3709
                000182DA   8F    08  AB  D1 00083 11$:     CMPL    8(CCB), #99034                       ; 3716
                                    14  12 0008B          BNEQ    12$                                  ;
                                    5B  DD 0008D          PUSHL   CCB                                  ; 3718
                00000000G  00       01  FB 0008F          CALLS   #1, SYS$WAIT                         ;
                                    5B  DD 00096          PUSHL   CCB                                  ; 3719
                00000000G  00       01  FB 00098          CALLS   #1, SYS$GET                          ;
                                    E2  11 0009F          BRB     11$                                  ; 3716
                           62       54  CC 000A1 12$:     XORL2   LOCK_FLAGS, (R2)                     ; 3729
                      22   AB    20  AB  B1 000A4          CMPW    32(CCB), 34(CCB)                     ; 3735
                                    1D  1B 000A9          BLEQU   13$                                  ;
                           19       08  AB  E9 000AB          BLBC    8(CCB), 13$                      ;
                           51       20  AB  3C 000AF          MOVZWL  32(CCB), R1                      ; 3738
                           50       22  AB  3C 000B3          MOVZWL  34(CCB), R0                      ;
                           51           50  C2 000B7          SUBL2   R0, R1                           ;
                           50       22  AB  3C 000BA          MOVZWL  34(CCB), R0                      ;
                           50       24  AB  C0 000BE          ADDL2   36(CCB), R0                      ;
        51          00     6E           00  2C 000C2          MOVC5   #0, (SP), #0, R1, (R0)           ;
                                    60      000C7                                                      ;
                      24   AB    9C  AB  D0 000C8 13$:     MOVL    -100(CCB), 36(CCB)                   ; 3743
                      20   AB        6E  B0 000CD          MOVW    SAVE_USZ, 32(CCB)                    ; 3744
                00000000'  EF    22  AB  3C 000D1          MOVZWL  34(CCB), RECOUNT                     ; 3745
                           0A    08  AB  E8 000D9          BLBS    8(CCB), 14$                          ; 3751
                           7E           01  CE 000DD          MNEGL   #1, -(SP)                        ;
                00000000G  00           01  FB 000E0          CALLS   #1, BAS$$STOP_IO                 ;
                      EC   AB    28  AB  D0 000E7 14$:     MOVL    40(CCB), -20(CCB)                    ; 3759
                           5E           04  C0 000EC          ADDL2   #4, SP                           ; 3761
                                    3C  BA 000EF          POPR    #^M<R2,R3,R4,R5>                     ;
                                        05 000F1          RSB                                          ;

; Routine Size:  242 bytes,    Routine Base:  _BAS$CODE + 0766

; 2490          3762  1
```

```
; 2492          3763  1 GLOBAL ROUTINE BAS$$REC_GRE (                    ! GET (relative) a record
; 2493          3764  1         FOREIGN_BUFFER, LOCK_FLAGS) : JSB_DO_READ NOVALUE =
; 2494          3765  1
; 2495          3766  1 !++
; 2496          3767  1 ! FUNCTIONAL DESCRIPTION:
; 2497          3768  1 !
; 2498          3769  1 !       Read one record from a relative file.  Modify the RAB if necessary for
; 2499          3770  1 !       foreign buffers.  Update RECOUNT if successful.  Otherwise, signal a fatal error.
; 2500          3771  1 !
; 2501          3772  1 ! FORMAL PARAMETERS:
; 2502          3773  1 !
; 2503          3774  1 !       FOREIGN_BUFFER.rl.v             The address of the CB of a foreign
; 2504          3775  1 !                                       buffer or 0
; 2505          3776  1 !       LOCK_FLAGS.rlu.v                bits to set in the RAB ROP to control
; 2506          3777  1 !                                       manual record locking (0 if none)
; 2507          3778  1 ! IMPLICIT INPUTS:
; 2508          3779  1 !
; 2509          3780  1 !       RAB$W_RSZ                       record size for foreign buffer
; 2510          3781  1 !       RAB$L_UBF                       buffer address for foreign buffer
; 2511          3782  1 !
; 2512          3783  1 ! IMPLICIT OUTPUTS:
; 2513          3784  1 !
; 2514          3785  1 !       RAB$B_RAC               Record access field
; 2515          3786  1 !       RECOUNT                 Own storage for RECOUNT function.
; 2516          3787  1 !       RAB$L_RBF               Record pointer in RAB.
; 2517          3788  1 !       RAB$W_USZ               record size of file buffer
; 2518          3789  1 !       RAB$L_UBF               address of buffer for file buffer
; 2519          3790  1 !
; 2520          3791  1 ! ROUTINE VALUE:
; 2521          3792  1 !
; 2522          3793  1 !       NONE
; 2523          3794  1 !
; 2524          3795  1 ! SIDE EFFECTS:
; 2525          3796  1 !
; 2526          3797  1 !       SIGNALS any RMS errors
; 2527          3798  1 !--
; 2528          3799  1
; 2529          3800  2     BEGIN
; 2530          3801  2
; 2531          3802  2     EXTERNAL REGISTER
; 2532          3803  2         CCB : REF BLOCK [, BYTE];
; 2533          3804  2
; 2534          3805  2     MAP
; 2535          3806  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2536          3807  2
; 2537          3808  2     LOCAL
; 2538          3809  2         RMS_STATUS,
; 2539          3810  2         SAVE_USZ;                               ! Save the USZ
; 2540          3811  2
; 2541          3812  2 !+
; 2542          3813  2 ! Save USZ in case it is modified.  It is faster to always
; 2543          3814  2 ! save and restore it, since that eliminates the test for foreign
; 2544          3815  2 ! buffers and single-character mode at the end of this routine.
; 2545          3816  2 !-
; 2546          3817  2     SAVE_USZ = .CCB [RAB$W_USZ];
; 2547          3818  2
; 2548          3819  2     !+
```

```
; 2549          3820  2      ! Set the record pointer field in the RAB to the user buffer.  This is
; 2550          3821  2      ! done on each element transmission and not just at OPEN because of RMS
; 2551          3822  2      ! Locate mode.
; 2552          3823  2      ! Fill the input buffer with Nulls.  Basic semantics require null fill if
; 2553          3824  2      ! the record read is shorter than the buffer.
; 2554          3825  2      ! Set the record access field in the RAB to sequential.  Perform the GET.
; 2555          3826  2      ! If RMS returns a failure status, signal the error.  If the GET is
; 2556          3827  2      ! successful, then update recount.
; 2557          3828  2      !-
; 2558          3829  2
; 2559          3830  2      IF .FOREIGN_BUFFER NEQ 0
; 2560          3831  2      THEN
; 2561          3832  3          BEGIN
; 2562          3833  3  !+
; 2563          3834  3  ! There is a foreign buffer.  Modify the file buffer to point to the
; 2564          3835  3  ! buffer associated with the foreign buffer's channel.
; 2565          3836  3  !-
; 2566          3837  3          CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
; 2567          3838  3          CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
; 2568          3839  3          END
; 2569          3840  2      ELSE
; 2570          3841  2          CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
; 2571          3842  2
; 2572          3843  2      CCB [RAB$B_RAC] = RAB$C_KEY;
; 2573          3844  2
; 2574          3845  2  !+
; 2575          3846  2  ! Set bits in RAB ROP without destroying ULK.
; 2576          3847  2  !-
; 2577          3848  2
; 2578          3849  2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 2579          3850  2
; 2580          3851  2      RMS_STATUS = $GET (RAB = .CCB);
; 2581          3852  2
; 2582          3853  2      IF .RMS_STATUS EQL RMS$_CONTROLC
; 2583          3854  2      THEN
; 2584          3855  2          BAS$$SIGNAL_CTRLC ();
; 2585          3856  2
; 2586          3857  2      IF NOT .RMS_STATUS
; 2587          3858  2      THEN
; 2588          3859  3          BEGIN
; 2589          3860  3  !+
; 2590          3861  3  ! We cannot call GET_ERROR because we must restore UBF and USZ.
; 2591          3862  3  !-
; 2592          3863  3
; 2593          3864  3          WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2594          3865  4              BEGIN
; 2595          3866  4              $WAIT (RAB = .CCB);
; 2596          3867  4              $GET (RAB = .CCB);
; 2597          3868  3              END;
; 2598          3869  3
; 2599          3870  2          END;
; 2600          3871  2
; 2601          3872  2  !+
; 2602          3873  2  ! Turn off bits in the RAB ROP so that subsequent I/O operations can not
; 2603          3874  2  ! inherit them.
; 2604          3875  2  !-
; 2605          3876  2
```

```
; 2606   3877  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 2607   3878  2
; 2608   3879  2 !+
; 2609   3880  2 ! If there are no errors, null pad the buffer.
; 2610   3881  2 !-
; 2611   3882  2
; 2612   3883  2     IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
; 2613   3884  2     THEN
; 2614   3885  2         CH$FILL (%X'00',
; 2615   3886  2             .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
; 2616   3887  2
; 2617   3888  2 !+
; 2618   3889  2 ! Before checking for errors, restore UBF and USZ, and set RECOUNT.
; 2619   3890  2 !-
; 2620   3891  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 2621   3892  2     CCB [RAB$W_USZ] = .SAVE_USZ;
; 2622   3893  2     RECOUNT = .CCB [RAB$W_RSZ];
; 2623   3894  2 !+
; 2624   3895  2 ! Any error remaining (which will be an error other than Record Stream
; 2625   3896  2 ! Active, RSA) is fatal.
; 2626   3897  2 !-
; 2627   3898  2
; 2628   3899  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 2629   3900  2
; 2630   3901  2 !+
; 2631   3902  2 ! Set LUB$A_RBUF_ADR to point to the buffer used by RMS.  It may move around
; 2632   3903  2 ! due to Locate Mode.
; 2633   3904  2 !-
; 2634   3905  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
; 2635   3906  2     RETURN;
; 2636   3907  1     END;                                          ! End of BAS$$REC_GRE


                                   3C  BB 00000 BAS$$REC_GRE::
                                                PUSHR   #^M<R2,R3,R4,R5>                          ; 3763
                       52          51  D0 00002         MOVL    R1, R2                            ;
                       7E     20   AB  3C 00005         MOVZWL  32(CCB), SAVE_USZ                 ; 3817
                                   50  D5 00009         TSTL    FOREIGN_BUFFER                    ; 3830
                                   1A  13 0000B         BEQL    1$                                ;
                       51     24   A0  D0 0000D         MOVL    36(FOREIGN_BUFFER), R1            ; 3837
                 24    AB          51  D0 00011         MOVL    R1, 36(CCB)                       ;
                 28    AB          51  D0 00015         MOVL    R1, 40(CCB)                       ;
                       50     20   A0  3C 00019         MOVZWL  32(FOREIGN_BUFFER), R0            ; 3838
                 20    AB          50  B0 0001D         MOVW    R0, 32(CCB)                       ;
                 22    AB          50  B0 00021         MOVW    R0, 34(CCB)                       ;
                                   05  11 00025         BRB     2$                                ; 3830
                 28    AB     24   AB  D0 00027 1$:     MOVL    36(CCB), 40(CCB)                  ; 3841
                 1E    AB          01  90 0002C 2$:     MOVB    #1, 30(CCB)                       ; 3843
                 04    AB          52  C8 00030         BISL2   LOCK_FLAGS, 4(CCB)                ; 3849
                                   5B  DD 00034         PUSHL   CCB                               ; 3851
           00000000G   00          01  FB 00036         CALLS   #1, SYS$GET                       ;
                       53          50  D0 0003D         MOVL    R0, RMS_STATUS                    ;
           00010651    8F          53  D1 00040         CMPL    RMS_STATUS, #67153                ; 3853
                                   07  12 00047         BNEQ    3$                                ;
```

```
                    00000000G  00          00  FB 00049          CALLS   #0, BAS$$SIGNAL_CTRLC                  ; 3855
                               1E          53  E8 00050 3$:      BLBS    RMS_STATUS, 5$                          ; 3857
                    000182DA   8F     08    AB  D1 00053 4$:      CMPL    8(CCB), #99034                          ; 3864
                                            14  12 0005B          BNEQ    5$                                      ;
                                            5B  DD 0005D          PUSHL   CCB                                     ; 3866
                    00000000G  00           01  FB 0005F          CALLS   #1, SYS$WAIT                            ;
                                            5B  DD 00066          PUSHL   CCB                                     ; 3867
                    00000000G  00           01  FB 00068          CALLS   #1, SYS$GET                             ;
                                            E2  11 0006F          BRB     4$                                      ; 3864
                          04   AB           52  CC 00071 5$:      XORL2   LOCK_FLAGS, 4(CCB)                      ; 3877
                          22   AB     20    AB  B1 00075          CMPW    32(CCB), 34(CCB)                        ; 3883
                                            1D  1B 0007A          BLEQU   6$                                      ;
                               19     08    AB  E9 0007C          BLBC    8(CCB), 6$                              ;
                               51     20    AB  3C 00080          MOVZWL  32(CCB), R1                             ; 3886
                               50     22    AB  3C 00084          MOVZWL  34(CCB), R0                             ;
                               51           50  C2 00088          SUBL2   R0, R1                                  ;
                               50     22    AB  3C 0008B          MOVZWL  34(CCB), R0                             ;
                               50     24    AB  C0 0008F          ADDL2   36(CCB), R0                             ;
          51          00       6E           00  2C 00093          MOVC5   #0, (SP), #0, R1, (R0)                  ;
                                            60     00098                                                          ;
                          24   AB     9C    AB  D0 00099 6$:      MOVL    -100(CCB), 36(CCB)                      ; 3891
                          20   AB           6E  B0 0009E          MOVW    SAVE_USZ, 32(CCB)                       ; 3892
                    00000000'  EF     22    AB  3C 000A2          MOVZWL  34(CCB), RECOUNT                        ; 3893
                               0A     08    AB  E8 000AA          BLBS    8(CCB), 7$                              ; 3899
                               7E           01  CE 000AE          MNEGL   #1, -(SP)                               ;
                    00000000G  00           01  FB 000B1          CALLS   #1, BAS$$STOP_IO                        ;
                          EC   AB     28    AB  D0 000B8 7$:      MOVL    40(CCB), -20(CCB)                       ; 3905
                               5E           04  C0 000BD          ADDL2   #4, SP                                  ; 3907
                                            3C  BA 000C0          POPR    #^M<R2,R3,R4,R5>                        ;
                                                05 000C2          RSB                                             ;

; Routine Size:  195 bytes,    Routine Base:  _BAS$CODE + 0858
```

```
; 2638          3908  1 GLOBAL ROUTINE BAS$$REC_GRFA (                    ! GET (by RFA) a record
; 2639          3909  1         FOREIGN_BUFFER, LOCK_FLAGS) : JSB_DO_READ NOVALUE =
; 2640          3910  1
; 2641          3911  1 !++
; 2642          3912  1 ! FUNCTIONAL DESCRIPTION:
; 2643          3913  1 !
; 2644          3914  1 !       Read one record from a file.  Modify the RAB if necessary for
; 2645          3915  1 !       foreign buffers.  Update RECOUNT if successful.  Otherwise, signal a fatal error.
; 2646          3916  1 !
; 2647          3917  1 ! FORMAL PARAMETERS:
; 2648          3918  1 !
; 2649          3919  1 !       FOREIGN_BUFFER.rl.v             The address of the CB of a foreign
; 2650          3920  1 !                                       buffer or 0
; 2651          3921  1 !       LOCK_FLAGS.rlu.v                bits to set in the RAB ROP to control
; 2652          3922  1 !                                       manual record locking (0 if none)
; 2653          3923  1 ! IMPLICIT INPUTS:
; 2654          3924  1 !
; 2655          3925  1 !       RAB$W_RSZ                       record size for foreign buffer
; 2656          3926  1 !       RAB$L_UBF                       buffer address for foreign buffer
; 2657          3927  1 !
; 2658          3928  1 ! IMPLICIT OUTPUTS:
; 2659          3929  1 !
; 2660          3930  1 !       RAB$B_RAC               Record access field
; 2661          3931  1 !       RECOUNT                 Own storage for RECOUNT function.
; 2662          3932  1 !       RAB$L_RBF               Record pointer in RAB.
; 2663          3933  1 !       RAB$W_USZ               record size of file buffer
; 2664          3934  1 !       RAB$L_UBF               address of buffer for file buffer
; 2665          3935  1 !
; 2666          3936  1 ! ROUTINE VALUE:
; 2667          3937  1 !
; 2668          3938  1 !       NONE
; 2669          3939  1 !
; 2670          3940  1 ! SIDE EFFECTS:
; 2671          3941  1 !
; 2672          3942  1 !       SIGNALs any RMS errors
; 2673          3943  1 !--
; 2674          3944  1
; 2675          3945  2     BEGIN
; 2676          3946  2
; 2677          3947  2     EXTERNAL REGISTER
; 2678          3948  2         CCB : REF BLOCK [, BYTE];
; 2679          3949  2
; 2680          3950  2     MAP
; 2681          3951  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2682          3952  2
; 2683          3953  2     LOCAL
; 2684          3954  2         RMS_STATUS,
; 2685          3955  2         SAVE_USZ;                               ! Save the USZ
; 2686          3956  2
; 2687          3957  2 !+
; 2688          3958  2 ! Save USZ in case it is modified.  It is faster to always
; 2689          3959  2 ! save and restore it, since that eliminates the test for foreign
; 2690          3960  2 ! buffers and single-character mode at the end of this routine.
; 2691          3961  2 !-
; 2692          3962  2     SAVE_USZ = .CCB [RAB$W_USZ];
; 2693          3963  2
; 2694          3964  2     !+
```

```
; 2695          3965  2      ! Set the record pointer field in the RAB to the user buffer.  This is
; 2696          3966  2      ! done on each element transmission and not just at OPEN because of RMS
; 2697          3967  2      ! Locate mode.
; 2698          3968  2      ! Fill the input buffer with Nulls.  Basic semantics require null fill if
; 2699          3969  2      ! the record read is shorter than the buffer.
; 2700          3970  2      ! Set the record access field in the RAB to sequential.  Perform the GET.
; 2701          3971  2      ! If RMS returns a failure status, signal the error.  If the GET is
; 2702          3972  2      ! successful, then update recount.
; 2703          3973  2      !-
; 2704          3974  2
; 2705          3975  2      IF .FOREIGN_BUFFER NEQ 0
; 2706          3976  2      THEN
; 2707          3977  3          BEGIN
; 2708          3978  3  !+
; 2709          3979  3  ! There is a foreign buffer.  Modify the file buffer to point to the
; 2710          3980  3  ! buffer associated with the foreign buffer's channel.
; 2711          3981  3  !-
; 2712          3982  3          CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
; 2713          3983  3          CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
; 2714          3984  3          END
; 2715          3985  2      ELSE
; 2716          3986  2          CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
; 2717          3987  2
; 2718          3988  2      CCB [RAB$B_RAC] = RAB$C_RFA;
; 2719          3989  2
; 2720          3990  2  !+
; 2721          3991  2  ! Set bits in RAB ROP without destroying ULK.
; 2722          3992  2  !-
; 2723          3993  2
; 2724          3994  2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 2725          3995  2
; 2726          3996  2      RMS_STATUS = $GET (RAB = .CCB);
; 2727          3997  2
; 2728          3998  2      IF .RMS_STATUS EQL RMS$_CONTROLC
; 2729          3999  2      THEN
; 2730          4000  2          BAS$$SIGNAL_CTRLC ();
; 2731          4001  2
; 2732          4002  2      IF NOT .RMS_STATUS
; 2733          4003  2      THEN
; 2734          4004  3          BEGIN
; 2735          4005  3  !+
; 2736          4006  3  ! We cannot call GET_ERROR because we must restore UBF and USZ.
; 2737          4007  3  !-
; 2738          4008  3
; 2739          4009  3          WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2740          4010  4              BEGIN
; 2741          4011  4              $WAIT (RAB = .CCB);
; 2742          4012  4              $GET (RAB = .CCB);
; 2743          4013  3              END;
; 2744          4014  3
; 2745          4015  2          END;
; 2746          4016  2
; 2747          4017  2  !+
; 2748          4018  2  ! Turn off bits in the RAB ROP so that subsequent I/O operations can not
; 2749          4019  2  ! inherit them.
; 2750          4020  2  !-
; 2751          4021  2
```

```
; 2752          4022  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 2753          4023  2
; 2754          4024  2 !+
; 2755          4025  2 ! If there are no errors, null pad the buffer.
; 2756          4026  2 !-
; 2757          4027  2
; 2758          4028  2     IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
; 2759          4029  2     THEN
; 2760          4030  2         CH$FILL (%X'00',
; 2761          4031  2             .CCB [RAB$W_USZ] - .CCB [RAB$w_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
; 2762          4032  2
; 2763          4033  2 !+
; 2764          4034  2 ! Before checking for errors, restore UBF and USZ, and set RECOUNT.
; 2765          4035  2 !-
; 2766          4036  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 2767          4037  2     CCB [RAB$W_USZ] = .SAVE_USZ;
; 2768          4038  2     RECOUNT = .CCB [RAB$W_RSZ];
; 2769          4039  2 !+
; 2770          4040  2 ! Any error remaining (which will be an error other than Record Stream
; 2771          4041  2 ! Active, RSA) is fatal.
; 2772          4042  2 !-
; 2773          4043  2
; 2774          4044  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 2775          4045  2
; 2776          4046  2 !+
; 2777          4047  2 ! Set LUB$A_RBUF_ADR to point to the buffer used by RMS.  It may move around
; 2778          4048  2 ! due to Locate Mode.
; 2779          4049  2 !-
; 2780          4050  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
; 2781          4051  2     RETURN;
; 2782          4052  1     END;                                        ! End of BAS$$REC_GRFA
```

```
                        3C  BB 00000 BAS$$REC_GRFA::
                                     PUSHR   #^M<R2,R3,R4,R5>                          ; 3908
          52            51  D0 00002         MOVL    R1, R2                            ;
          7E      20    AB  3C 00005         MOVZWL  32(CCB), SAVE_USZ                 ; 3962
                        50  D5 00009         TSTL    FOREIGN_BUFFER                    ; 3975
                        1A  13 0000B         BEQL    1$                                ;
          51      24    A0  D0 0000D         MOVL    36(FOREIGN_BUFFER), R1            ; 3982
    24    AB            51  D0 00011         MOVL    R1, 36(CCB)                       ;
    28    AB            51  D0 00015         MOVL    R1, 40(CCB)                       ;
          50      20    A0  3C 00019         MOVZWL  32(FOREIGN_BUFFER), R0            ; 3983
    20    AB            50  B0 0001D         MOVW    R0, 32(CCB)                       ;
    22    AB            50  B0 00021         MOVW    R0, 34(CCB)                       ;
                        05  11 00025         BRB     2$                                ; 3975
    28    AB      24    AB  D0 00027 1$:     MOVL    36(CCB), 40(CCB)                  ; 3986
    1E    AB            C2  90 0002C 2$:     MOVB    #2, 30(CCB)                       ; 3988
    04    AB            52  C8 00030         BISL2   LOCK_FLAGS, 4(CCB)                ; 3994
                        5B  DD 00034         PUSHL   CCB                               ; 3996
000000006   00          01  FB 00036         CALLS   #1, SYS$GET                       ;
          53            50  D0 0003D         MOVL    R0, RMS_STATUS                    ;
00010651  8F            53  D1 00040         CMPL    RMS_STATUS, #67153                ; 3998
                        07  12 00047         BNEQ    3$                                ;
```

```
                 00000000G  00          00  FB 00049          CALLS   #0, BAS$$SIGNAL_CTRLC                  ; 4000
                            1E          53  E8 00050 3$:      BLBS    RMS_STATUS, 5$                         ; 4002
                 000182DA   8F     08    AB  D1 00053 4$:      CMPL    8(CCB), #99034                         ; 4009
                                         14  12 0005B          BNEQ    5$                                     ;
                                         5B  DD 0005D          PUSHL   CCB                                    ; 4011
                 00000000G  00           01  FB 0005F          CALLS   #1, SYS$WAIT                           ;
                                         5B  DD 00066          PUSHL   CCB                                    ; 4012
                 00000000G  00           01  FB 00068          CALLS   #1, SYS$GET                            ;
                                         E2  11 0006F          BRB     4$                                     ; 4009
                      04    AB           52  CC 00071 5$:      XORL2   LOCK_FLAGS, 4(CCB)                     ; 4022
                      22    AB     20    AB  B1 00075          CMPW    32(CCB), 34(CCB)                       ; 4028
                                         1D  1B 0007A          BLEQU   6$                                     ;
                            19     08    AB  E9 0007C          BLBC    8(CCB), 6$                             ;
                            51     20    AB  3C 00080          MOVZWL  32(CCB), R1                            ; 4031
                            50     22    AB  3C 00084          MOVZWL  34(CCB), R0                            ;
                            51           50  C2 00088          SUBL2   R0, R1                                 ;
                            50     22    AB  3C 0008B          MOVZWL  34(CCB), R0                            ;
                            50     24    AB  C0 0008F          ADDL2   36(CCB), R0                            ;
        51          00      6E           00  2C 00093          MOVC5   #0, (SP), #0, R1, (R0)                 ;
                                         60     00098                                                        ;
                      24    AB     9C    AB  D0 00099 6$:      MOVL    -100(CCB), 36(CCB)                     ; 4036
                      20    AB           6E  B0 0009E          MOVW    SAVE_USZ, 32(CCB)                      ; 4037
                 00000000'  EF     22    AB  3C 000A2          MOVZWL  34(CCB), RECOUNT                       ; 4038
                            0A     08    AB  E8 000AA          BLBS    8(CCB), 7$                             ; 4044
                            7E           01  CE 000AE          MNEGL   #1, -(SP)                              ;
                 00000000G  00           01  FB 000B1          CALLS   #1, BAS$$STOP_IO                       ;
                      EC    AB     28    AB  D0 000B8 7$:      MOVL    40(CCB), -20(CCB)                      ; 4050
                            5E           04  C0 000BD          ADDL2   #4, SP                                 ; 4052
                                         3C  BA 000C0          POPR    #^M<R2,R3,R4,R5>                       ;
                                             05 000C2          RSB                                            ;

; Routine Size:  195 bytes,    Routine Base:  _BAS$CODE + 091B

; 2783          4053  1
; 2784          4054  1
```

```
; 2786          4055  1 GLOBAL ROUTINE BAS$$REC_PSE (                    ! PUT (sequential) a record
; 2787          4056  1         COUNT,                                   ! No. of bytes to write
; 2788          4057  1         FOREIGN_BUFFER                           ! pointer to foreign buffer CB or 0
; 2789          4058  1     ) : JSB_PUT NOVALUE =
; 2790          4059  1
; 2791          4060  1 !++
; 2792          4061  1 ! FUNCTIONAL DESCRIPTION:
; 2793          4062  1 !
; 2794          4063  1 !       Check for "foreign buffers" and point RAB$L_RSZ to foreign USZ if there.
; 2795          4064  1 !       Write one record.  If successful then return; otherwise, signal a fatal
; 2796          4065  1 !       error.
; 2797          4066  1 !
; 2798          4067  1 ! FORMAL PARAMETERS:
; 2799          4068  1 !
; 2800          4069  1 !       COUNT.rl.v              No. of bytes to write
; 2801          4070  1 !       FOREIGN_BUFFER.rl.v     pointer to foreign buffer CB or 0
; 2802          4071  1 !
; 2803          4072  1 ! IMPLICIT INPUTS:
; 2804          4073  1 !
; 2805          4074  1 !       RAB$W_RSZ               of foreign buffer
; 2806          4075  1 !       RAB$L_RBF               of foreign buffer
; 2807          4076  1 !       LUB$V_CCO               Cancel control O
; 2808          4077  1 !
; 2809          4078  1 ! IMPLICIT OUTPUTS:
; 2810          4079  1 !
; 2811          4080  1 !       RAB$L_RBF               for "file" buffer
; 2812          4081  1 !       RAB$W_RSZ               length of record to write
; 2813          4082  1 !       LUB$L_LOG_RECNO         logical record number
; 2814          4083  1 !       RAB$B_RAC               record access field
; 2815          4084  1 !       RAB$V_CCO               Cancel control O
; 2816          4085  1 !
; 2817          4086  1 ! ROUTINE VALUE:
; 2818          4087  1 !
; 2819          4088  1 !       NONE
; 2820          4089  1 !
; 2821          4090  1 ! SIDE EFFECTS:
; 2822          4091  1 !
; 2823          4092  1 !       SIGNALs any RMS errors
; 2824          4093  1 !--
; 2825          4094  1
; 2826          4095  2     BEGIN
; 2827          4096  2
; 2828          4097  2     EXTERNAL REGISTER
; 2829          4098  2         CCB : REF BLOCK [, BYTE];
; 2830          4099  2
; 2831          4100  2     LOCAL
; 2832          4101  2         RMS_STATUS;
; 2833          4102  2
; 2834          4103  2     MAP
; 2835          4104  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2836          4105  2
; 2837          4106  2 !+
; 2838          4107  2 ! Copy the current status of the cancel-control-o bit in the LUB
; 2839          4108  2 ! (possibly set by RCTRLO) into the RAB, and clear it from the
; 2840          4109  2 ! LUB.  The net effect of this is that if the bit is set in the
; 2841          4110  2 ! LUB, then the CANCTRLO modifier will be applied to this write
; 2842          4111  2 ! operation only.
```

```
; 2843          4112  2 !-
; 2844          4113  2
; 2845          4114  2     CCB [RAB$V_CCO] = .CCB [LUB$V_CCO];
; 2846          4115  2     CCB [LUB$V_CCO] = 0;
; 2847          4116  2
; 2848          4117  2     !+
; 2849          4118  2     ! Set the recordsize field in the RAB based on COUNT.
; 2850          4119  2     ! Set the record address field in the RAB to the user buffer.
; 2851          4120  2     ! Perform the PUT.
; 2852          4121  2     ! If RMS returns a failure status, signal the error.
; 2853          4122  2     !-
; 2854          4123  2
; 2855          4124  2     CCB [RAB$W_RSZ] = .COUNT;
; 2856          4125  2     CCB [RAB$B_RAC] = (IF .CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX THEN RAB$C_KEY ELSE RAB$C_SEQ);
; 2857          4126  2
; 2858          4127  2     IF .FOREIGN_BUFFER NEQA 0
; 2859          4128  2     THEN
; 2860          4129  2 !+
; 2861          4130  2 ! There is a foreign buffer.  Point RAB$L_UBF to it.
; 2862          4131  2 !-
; 2863          4132  2         CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF]
; 2864          4133  2     ELSE
; 2865          4134  2         CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
; 2866          4135  2
; 2867          4136  2     RMS_STATUS = $PUT (RAB = .CCB);
; 2868          4137  2
; 2869          4138  2     IF .RMS_STATUS EQL RMS$_CONTROLC
; 2870          4139  2     THEN
; 2871          4140  2         BAS$$SIGNAL_CTRLC ();
; 2872          4141  2
; 2873          4142  2     IF NOT .RMS_STATUS
; 2874          4143  2     THEN
; 2875          4144  3         BEGIN
; 2876          4145  3 !+
; 2877          4146  3 ! We cannot call PUT_ERROR because we must restore UBF and USZ.
; 2878          4147  3 !-
; 2879          4148  3
; 2880          4149  3         WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2881          4150  4             BEGIN
; 2882          4151  4             $WAIT (RAB = .CCB);
; 2883          4152  4             $PUT (RAB = .CCB);
; 2884          4153  3             END;
; 2885          4154  3
; 2886          4155  2         END;
; 2887          4156  2
; 2888          4157  2 !+
; 2889          4158  2 ! Restore RAB$L_UBF in case there was a foreign buffer.
; 2890          4159  2 !-
; 2891          4160  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 2892          4161  2 !+
; 2893          4162  2 ! Point LUB$A_RBUF_PTR off to the buffer used by RMS.
; 2894          4163  2 !-
; 2895          4164  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
; 2896          4165  2 !+
; 2897          4166  2 ! Any error remaining (which will be an error other than Record Stream
; 2898          4167  2 ! Active, RSA) is fatal.
; 2899          4168  2 !-
```

```
; 2900          4169  2
; 2901          4170  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 2902          4171  2
; 2903          4172  2     RETURN;
; 2904          4173  1     END;                                        ' End of BAS$$REC_PSE


                                    5E          04  C2 00000 BAS$$REC_PSE::
                                                             SUBL2   #4, SP                          ; 4055
        7E      A0  AB              01          02  EF 00003         EXTZV   #2, #1, -96(CCB), -(SP)         ; 4114
  07    AB          01              07          8E  F0 00009         INSV    (SP)+, #7, #1, 7(CCB)           ;
                              A0    AB          04  8A 0000F         BICB2   #4, -96(CCB)                    ; 4115
                              22    AB          51  B0 00013         MOVW    COUNT, 34(CCB)                  ; 4124
                                    03      C4  AB  91 00017         CMPB    -60(CCB), #3                    ; 4125
                                                05  12 0001B         BNEQ    1$                              ;
                                    51          01  D0 0001D         MOVL    #1, R1                          ;
                                                02  11 00020         BRB     2$                              ;
                                                51  D4 00022 1$:     CLRL    R1                              ;
                              1E    AB          51  90 00024 2$:     MOVB    R1, 30(CCB)                     ;
                                                50  D5 00028         TSTL    FOREIGN_BUFFER                  ; 4127
                                                0E  13 0002A         BEQL    3$                              ;
                                    50      24  A0  D0 0002C         MOVL    36(FOREIGN_BUFFER), R0          ; 4132
                              24    AB          50  D0 00030         MOVL    R0, 36(CCB)                     ;
                              28    AB          50  D0 00034         MOVL    R0, 40(CCB)                     ;
                                                05  11 00038         BRB     4$                              ;
                              28    AB      24  AB  D0 0003A 3$:     MOVL    36(CCB), 40(CCB)                ; 4134
                                                5B  DD 0003F 4$:     PUSHL   CCB                             ; 4136
                        00000000G   00          01  FB 00041         CALLS   #1, SYS$PUT                     ;
                                    6E          50  D0 00048         MOVL    R0, RMS_STATUS                  ;
                        00010651    8F          6E  D1 0004B         CMPL    RMS_STATUS, #67153              ; 4138
                                                07  12 00052         BNEQ    5$                              ;
                        00000000G   00          00  FB 00054         CALLS   #0, BAS$$SIGNAL_CTRLC           ; 4140
                                    1E          6E  E8 0005B 5$:     BLBS    RMS_STATUS, 7$                  ; 4142
                        000182DA    8F      08  AB  D1 0005E 6$:     CMPL    8(CCB), #99034                  ; 4149
                                                14  12 00066         BNEQ    7$                              ;
                                                5B  DD 00068         PUSHL   CCB                             ; 4151
                        00000000G   00          01  FB 0006A         CALLS   #1, SYS$WAIT                    ;
                                                5B  DD 00071         PUSHL   CCB                             ; 4152
                        00000000G   00          01  FB 00073         CALLS   #1, SYS$PUT                     ;
                                                E2  11 0007A         BRB     6$                              ; 4149
                              24    AB      9C  AB  D0 0007C 7$:     MOVL    -100(CCB), 36(CCB)              ; 4160
                              EC    AB      24  AB  D0 00081         MOVL    36(CCB), -20(CCB)               ; 4164
                                    0A      08  AB  E8 00086         BLBS    8(CCB), 8$                      ; 4170
                                    7E          01  CE 0008A         MNEGL   #1, -(SP)                       ;
                        00000000G   00          01  FB 0008D         CALLS   #1, BAS$$STOP_IO                ;
                                    5E          04  C0 00094 8$:     ADDL2   #4, SP                          ; 4173
                                                    05 00097         RSB                                     ;

; Routine Size:  152 bytes,    Routine Base:  _BAS$CODE + 09DE

; 2905          4174  1
```

```
; 2907          4175  1 GLOBAL ROUTINE BAS$$REC_PRE (                    ! PUT (relative) a record
; 2908          4176  1         COUNT,                                   ! No. of bytes to write
; 2909          4177  1         FOREIGN_BUFFER                           ! pointer to foreign CB or 0
; 2910          4178  1     ) : JSB_PUT NOVALUE -
; 2911          4179  1
; 2912          4180  1 !++
; 2913          4181  1 ! FUNCTIONAL DESCRIPTION:
; 2914          4182  1 !
; 2915          4183  1 !       Check for a foreign buffer and point to it if necessary.
; 2916          4184  1 !       Write one record.  If successful then return; otherwise, signal a fatal
; 2917          4185  1 !       error.
; 2918          4186  1 !
; 2919          4187  1 ! FORMAL PARAMETERS:
; 2920          4188  1 !
; 2921          4189  1 !       COUNT.rl.v              No. of bytes to write
; 2922          4190  1 !       FOREIGN_BUFFER.rl.v     pointer to foreign CB or 0
; 2923          4191  1 !
; 2924          4192  1 ! IMPLICIT INPUTS:
; 2925          4193  1 !
; 2926          4194  1 !       RAB$L_UBF               for the foreign buffer (buffer pointer)
; 2927          4195  1 !       RAB$W_USZ               for the foreign buffer (buffer size)
; 2928          4196  1 !
; 2929          4197  1 ! IMPLICIT OUTPUTS:
; 2930          4198  1 !
; 2931          4199  1 !       RAB$W_RSZ               length of record to write (file buffer)
; 2932          4200  1 !       RAB$L_RBF               pointer to file CB
; 2933          4201  1 !       LUB$L_LOG_RECNO         logical record number
; 2934          4202  1 !       RAB$B_RAC               record access field
; 2935          4203  1 !
; 2936          4204  1 ! ROUTINE VALUE:
; 2937          4205  1 !
; 2938          4206  1 !       NONE
; 2939          4207  1 !
; 2940          4208  1 ! SIDE EFFECTS:
; 2941          4209  1 !
; 2942          4210  1 !       SIGNALs any RMS errors
; 2943          4211  1 !--
; 2944          4212  1
; 2945          4213  2     BEGIN
; 2946          4214  2
; 2947          4215  2     EXTERNAL REGISTER
; 2948          4216  2         CCB : REF BLOCK [, BYTE];
; 2949          4217  2
; 2950          4218  2     LOCAL
; 2951          4219  2         RMS_STATUS;
; 2952          4220  2
; 2953          4221  2     MAP
; 2954          4222  2         FOREIGN_BUFFER : REF BLOCK [, BYTE];
; 2955          4223  2
; 2956          4224  2     !+
; 2957          4225  2     ! Set the recordsize field in the RAB based on COUNT.
; 2958          4226  2     ! Set the record address field in the RAB to the user buffer.
; 2959          4227  2     ! Set the record access field in the RAB to relative.  Perform the PUT.
; 2960          4228  2     ! If RMS returns a failure status, signal the error.
; 2961          4229  2     !-
; 2962          4230  2
; 2963          4231  2     CCB [RAB$W_RSZ] = .COUNT;
```

```
; 2964          4232  2     CCB [RAB$B_RAC] = RAB$C_KEY;
; 2965          4233  2
; 2966          4234  2     IF .FOREIGN_BUFFER NEQA 0
; 2967          4235  2     THEN
; 2968          4236  2 !+
; 2969          4237  2 ! There is a foreign buffer.  Point off to the buffer but don't do the
; 2970          4238  2 ! size.  A PUT with count would not work right, so the size is passed in.
; 2971          4239  2 !-
; 2972          4240  2         CCB [RAB$L_UBF] = CCB [RAB$L_RBF] = .FOREIGN_BUFFER [RAB$L_UBF]
; 2973          4241  2     ELSE
; 2974          4242  2         CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
; 2975          4243  2
; 2976          4244  2     RMS_STATUS = $PUT (RAB = .CCB);
; 2977          4245  2
; 2978          4246  2     IF .RMS_STATUS EQL RMS$_CONTROLC
; 2979          4247  2     THEN
; 2980          4248  2         BAS$$SIGNAL_CTRLC ();
; 2981          4249  2
; 2982          4250  2     IF NOT .RMS_. ATUS
; 2983          4251  2     THEN
; 2984          4252  3         BEGIN
; 2985          4253  3 !+
; 2986          4254  3 ! We cannot call GET_ERROR because we must restore UBF and USZ.
; 2987          4255  3 !-
; 2988          4256  3
; 2989          4257  3         WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
; 2990          4258  4             BEGIN
; 2991          4259  4             $WAIT (RAB = .CCB);
; 2992          4260  4             $PUT (RAB = .CCB);
; 2993          4261  3             END;
; 2994          4262  3
; 2995          4263  2         END;
; 2996          4264  2
; 2997          4265  2 !+
; 2998          4266  2 ! Restore RAB$L_UBF in case there was a foreign buffer.
; 2999          4267  2 !-
; 3000          4268  2     CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
; 3001          4269  2 !+
; 3002          4270  2 ! Point LUB$A_BUF_PTR off to the buffer used by RMS.
; 3003          4271  2 !-
; 3004          4272  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
; 3005          4273  2 !+
; 3006          4274  2 ! Any error remaining (which will be an error other than Record Stream
; 3007          4275  2 ! Active, RSA) is fatal.
; 3008          4276  2 !-
; 3009          4277  2
; 3010          4278  2     IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3011          4279  2
; 3012          4280  2     RETURN;
; 3013          4281  1     END;                                        ! End of BAS$$REC_PRE


                5E          04  C2 00000 BAS$$REC_PRE::
                                         SUBL2   #4, SP                                  ; 4175
```

```
                22   AB          51  B0 00003          MOVW    COUNT, 34(CCB)                          ; 4231
                1E   AB          01  90 00007          MOVB    #1, 30(CCB)                             ; 4232
                                 50  D5 0000B          TSTL    FOREIGN_BUFFER                          ; 4234
                                 0E  13 0000D          BEQL    1$                                      ;
                     50      24  A0  D0 0000F          MOVL    36(FOREIGN_BUFFER), R0                  ; 4240
                28   AB          50  D0 00013          MOVL    R0, 40(CCB)                             ;
                24   AB          50  D0 00017          MOVL    R0, 36(CCB)                             ;
                                 05  11 0001B          BRB     2$                                      ;
                28   AB      24  AB  D0 0001D 1$:      MOVL    36(CCB), 40(CCB)                        ; 4242
                                 5B  DD 00022 2$:      PUSHL   CCB                                     ; 4244
           00000000G 00          01  FB 00024          CALLS   #1, SYS$PUT                             ;
                     6E          50  D0 0002B          MOVL    R0, RMS_STATUS                          ;
           00010651  8F          6E  D1 0002E          CMPL    RMS_STATUS, #67153                      ; 4246
                                 07  12 00035          BNEQ    3$                                      ;
           00000000G 00          00  FB 00037          CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 4248
                     1E          6E  E8 0003E 3$:      BLBS    RMS_STATUS, 5$                          ; 4250
           000182DA  8F      08  AB  D1 00041 4$:      CMPL    8(CCB), #99034                          ; 4257
                                 14  12 00049          BNEQ    5$                                      ;
                                 5B  DD 0004B          PUSHL   CCB                                     ; 4259
           00000000G 00          01  FB 0004D          CALLS   #1, SYS$WAIT                            ;
                                 5B  DD 00054          PUSHL   CCB                                     ; 4260
           00000000G 00          01  FB 00056          CALLS   #1, SYS$PUT                             ;
                                 E2  11 0005D          BRB     4$                                      ; 4257
                24   AB      9C  AB  D0 0005F 5$:      MOVL    -100(CCB), 36(CCB)                      ; 4268
                EC   AB      24  AB  D0 00064          MOVL    36(CCB), -20(CCB)                       ; 4272
                     0A      08  AB  E8 00069          BLBS    8(CCB), 6$                              ; 4278
                     7E          01  CE 0006D          MNEGL   #1, -(SP)                               ;
           00000000G 00          01  FB 00070          CALLS   #1, BAS$$STOP_IO                        ;
                     5E          04  C0 00077 6$:      ADDL2   #4, SP                                  ; 4281
                                     05 0007A          RSB                                             ;

; Routine Size:  123 bytes,    Routine Base:  _BAS$CODE + 0A76

; 3014          4282  1
```

```
; 3016          4283  1 GLOBAL ROUTINE BAS$$REC_FSE (                    ! FIND (sequential) a record
; 3017          4284  1         LOCK_FLAGS
; 3018          4285  1   ) : JSB_REC2 NOVALUE =
; 3019          4286  1
; 3020          4287  1 !++
; 3021          4288  1 ! FUNCTIONAL DESCRIPTION:
; 3022          4289  1 !
; 3023          4290  1 !       Find next record.  If successful then return; otherwise, signal a fatal
; 3024          4291  1 !       error.
; 3025          4292  1 !
; 3026          4293  1 ! FORMAL PARAMETERS:
; 3027          4294  1 !
; 3028          4295  1 !       LOCK_FLAGS.rlu.v        bits to set in the RAB ROP to control manual
; 3029          4296  1 !                               record locking (0 if none)
; 3030          4297  1 !
; 3031          4298  1 ! IMPLICIT INPUTS:
; 3032          4299  1 !
; 3033          4300  1 !       NONE
; 3034          4301  1 !
; 3035          4302  1 ! IMPLICIT OUTPUTS:
; 3036          4303  1 !
; 3037          4304  1 !       RAB$B_RAC               record access field
; 3038          4305  1 !
; 3039          4306  1 ! ROUTINE VALUE:
; 3040          4307  1 !
; 3041          4308  1 !       NONE
; 3042          4309  1 !
; 3043          4310  1 ! SIDE EFFECTS:
; 3044          4311  1 !
; 3045          4312  1 !       Finds next record in file on this logical unit.
; 3046          4313  1 !       SIGNALs any RMS errors
; 3047          4314  1 !--
; 3048          4315  1
; 3049          4316  2     BEGIN
; 3050          4317  2
; 3051          4318  2     EXTERNAL REGISTER
; 3052          4319  2         CCB : REF BLOCK [, BYTE];
; 3053          4320  2
; 3054          4321  2     LOCAL
; 3055          4322  2         RMS_STATUS;
; 3056          4323  2
; 3057          4324  2     !+
; 3058          4325  2     ! Set the record access field in the RAB to sequential.  Perform the FIND.
; 3059          4326  2     ! If RMS returns a failure status, signal the error.
; 3060          4327  2     !-
; 3061          4328  2
; 3062          4329  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3063          4330  2
; 3064          4331  2     !+
; 3065          4332  2     ! Set bits in RAB ROP without clearing ULK.
; 3066          4333  2     !-
; 3067          4334  2
; 3068          4335  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 3069          4336  2
; 3070          4337  2     !+
; 3071          4338  2     ! perform the FIND.
; 3072          4339  2     !-
```

```
; 3073          4340  2
; 3074          4341  2     RMS_STATUS = $FIND (RAB = .CCB);
; 3075          4342  2
; 3076          4343  2     !+
; 3077          4344  2     ! Turn off bits so that subsequent operations will not inherit them.
; 3078          4345  2     !-
; 3079          4346  2
; 3080          4347  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 3081          4348  2
; 3082          4349  2     !+
; 3083          4350  2     ! signal if the FIND failed.
; 3084          4351  2     !-
; 3085          4352  2     IF NOT .RMS_STATUS
; 3086          4353  2     THEN
; 3087          4354  2         BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3088          4355  2
; 3089          4356  2     RETURN;
; 3090          4357  1     END;                                        ! End of BAS$$REC_FSE


                                                        .EXTRN  SYS$FIND

                                    52  DD 00000 BAS$$REC_FSE::
                                                        PUSHL   R2                                  ; 4283
                 52                 50  D0 00002        MOVL    R0, R2                              ;
                            1E      AB  94 00005        CLRB    30(CCB)                             ; 4329
            04   AB                 52  C8 00008        BISL2   LOCK_FLAGS, 4(CCB)                  ; 4335
                                    5B  DD 0000C        PUSHL   CCB                                 ; 4341
     00000000G   00                 01  FB 0000E        CALLS   #1, SYS$FIND                        ;
            04   AB                 52  CC 00015        XORL2   LOCK_FLAGS, 4(CCB)                  ; 4347
                 0A                 50  E8 00019        BLBS    RMS_STATUS, 1$                      ; 4352
                 7E                 01  CE 0001C        MNEGL   #1, -(SP)                           ; 4354
     00000000G   00                 01  FB 0001F        CALLS   #1, BAS$$STOP_IO                    ;
                                    04  BA 00026 1$:    POPR    #^M<R2>                             ; 4357
                                        05 00028        RSB                                         ;

; Routine Size:  41 bytes,    Routine Base:  _BAS$CODE + 0AF1
```

```
; 3092          4358  1 GLOBAL ROUTINE BAS$$REC_FRFA (                    ! FIND (by RFA) a record
; 3093          4359  1         LOCK_FLAGS
; 3094          4360  1   ) : JSB_REC2 NOVALUE =
; 3095          4361  1
; 3096          4362  1 !++
; 3097          4363  1 ! FUNCTIONAL DESCRIPTION:
; 3098          4364  1 !
; 3099          4365  1 !       Find record by RFA stored in the RAB.  If successful then return; otherwise, signal a fatal
; 3100          4366  1 !       error.
; 3101          4367  1 !
; 3102          4368  1 ! FORMAL PARAMETERS:
; 3103          4369  1 !
; 3104          4370  1 !       LOCK_FLAGS.rlu.v        bits to set in the RAB ROP to control manual
; 3105          4371  1 !                               record locking (0 if none)
; 3106          4372  1 !
; 3107          4373  1 ! IMPLICIT INPUTS:
; 3108          4374  1 !
; 3109          4375  1 !       NONE
; 3110          4376  1 !
; 3111          4377  1 ! IMPLICIT OUTPUTS:
; 3112          4378  1 !
; 3113          4379  1 !       RAB$B_RAC               record access field
; 3114          4380  1 !
; 3115          4381  1 ! ROUTINE VALUE:
; 3116          4382  1 !
; 3117          4383  1 !       NONE
; 3118          4384  1 !
; 3119          4385  1 ! SIDE EFFECTS:
; 3120          4386  1 !
; 3121          4387  1 !       finds next record in file on this logical unit.
; 3122          4388  1 !       SIGNALs any RMS errors
; 3123          4389  1 !--
; 3124          4390  1
; 3125          4391  2     BEGIN
; 3126          4392  2
; 3127          4393  2     EXTERNAL REGISTER
; 3128          4394  2         CCB : REF BLOCK [, BYTE];
; 3129          4395  2
; 3130          4396  2     LOCAL
; 3131          4397  2         RMS_STATUS;
; 3132          4398  2
; 3133          4399  2     !+
; 3134          4400  2     ! Set the record access field in the RAB to sequential.  Perform the FIND.
; 3135          4401  2     ! If RMS returns a failure status, signal the error.
; 3136          4402  2     !-
; 3137          4403  2
; 3138          4404  2     CCB [RAB$B_RAC] = RAB$C_RFA;
; 3139          4405  2
; 3140          4406  2     !+
; 3141          4407  2     ! Set bits in RAB ROP without clearing ULK.
; 3142          4408  2     !-
; 3143          4409  2
; 3144          4410  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 3145          4411  2
; 3146          4412  2     !+
; 3147          4413  2     ! perform the FIND.
; 3148          4414  2     !-
```

```
; 3149          4415  2
; 3150          4416  2     RMS_STATUS = $FIND (RAB = .CCB);
; 3151          4417  2
; 3152          4418  2     !+
; 3153          4419  2     ! Turn off bits so that subsequent operations will not inherit them.
; 3154          4420  2     !-
; 3155          4421  2
; 3156          4422  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 3157          4423  2
; 3158          4424  2     !+
; 3159          4425  2     ! Signal if the FIND failed.
; 3160          4426  2     !-
; 3161          4427  2     IF NOT .RMS_STATUS
; 3162          4428  2     THEN
; 3163          4429  2         BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3164          4430  2
; 3165          4431  2     RETURN;
; 3166          4432  1     END;                                        ! End of BAS$$REC_FRFA


                                  52  DD 00000 BAS$$REC_FRFA::
                                                  PUSHL   R2                                  ; 4358
                        52        50  D0 00002          MOVL    R0, R2                          ;
                  1E    AB        02  90 00005          MOVB    #2, 30(CCB)                     ; 4404
                  04    AB        52  C8 00009          BISL2   LOCK_FLAGS, 4(CCB)              ; 4410
                                  5B  DD 0000D          PUSHL   CCB                             ; 4416
            00000000G   00        01  FB 0000F          CALLS   #1, SYS$FIND                    ;
                  04    AB        52  CC 00016          XORL2   LOCK_FLAGS, 4(CCB)              ; 4422
                        0A        50  E8 0001A          BLBS    RMS_STATUS, 1$                  ; 4427
                        7E        01  CE 0001D          MNEGL   #1, -(SP)                       ; 4429
            00000000G   00        01  FB 00020          CALLS   #1, BAS$$STOP_IO                ;
                                  04  BA 00027 1$:      POPR    #^M<R2>                         ; 4432
                                      05 00029          RSB                                     ;

; Routine Size:  42 bytes,    Routine Base:  _BAS$CODE + 0B1A

; 3167          4433  1
; 3168          4434  1
```

```
; 3170          4435  1 GLOBAL ROUTINE BAS$$REC_FRE (                    ! FIND (relative) a record
; 3171          4436  1         LOCK_FLAGS
; 3172          4437  1   ) : JSB_REC0 NOVALUE =
; 3173          4438  1
; 3174          4439  1 !++
; 3175          4440  1 ! FUNCTIONAL DESCRIPTION:
; 3176          4441  1 !
; 3177          4442  1 !       Find next record.  If successful then return; otherwise, signal a fatal
; 3178          4443  1 !       error.
; 3179          4444  1 !
; 3180          4445  1 ! FORMAL PARAMETERS:
; 3181          4446  1 !
; 3182          4447  1 !       LOCK_FLAGS.rlu.v        bits to set in the RAB ROP to control manual
; 3183          4448  1 !                               record locking (0 if none)
; 3184          4449  1 !
; 3185          4450  1 ! IMPLICIT INPUTS:
; 3186          4451  1 !
; 3187          4452  1 !       NONE
; 3188          4453  1 !
; 3189          4454  1 ! IMPLICIT OUTPUTS:
; 3190          4455  1 !
; 3191          4456  1 !       RAB$B_RAC               record access field
; 3192          4457  1 !
; 3193          4458  1 ! ROUTINE VALUE:
; 3194          4459  1 !
; 3195          4460  1 !       NONE
; 3196          4461  1 !
; 3197          4462  1 ! SIDE EFFECTS:
; 3198          4463  1 !
; 3199          4464  1 !       SIGNALs any RMS errors
; 3200          4465  1 !--
; 3201          4466  1
; 3202          4467  2     BEGIN
; 3203          4468  2
; 3204          4469  2     EXTERNAL REGISTER
; 3205          4470  2         CCB : REF BLOCK [, BYTE];
; 3206          4471  2
; 3207          4472  2     LOCAL
; 3208          4473  2         RMS_STATUS;
; 3209          4474  2
; 3210          4475  2     !+
; 3211          4476  2     ! Set the record access field in the RAB to sequential.  Perform the FIND.
; 3212          4477  2     ! If RMS returns a failure status, signal the error.
; 3213          4478  2     !-
; 3214          4479  2
; 3215          4480  2     CCB [RAB$B_RAC] = RAB$C_KEY;
; 3216          4481  2
; 3217          4482  2     !+
; 3218          4483  2     ! Set bits in the RAB ROP without clearing ULK.
; 3219          4484  2     !-
; 3220          4485  2
; 3221          4486  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 3222          4487  2
; 3223          4488  2     !+
; 3224          4489  2     ! perform the FIND.
; 3225          4490  2     !-
; 3226          4491  2
```

```
; 3227          4492  2     RMS_STATUS = $FIND (RAB = .CCB);
; 3228          4493  2
; 3229          4494  2     !+
; 3230          4495  2     ! Turn off bits in the RAB ROP so that subsequent operations do not
; 3231          4496  2     ! inherit them.
; 3232          4497  2     !-
; 3233          4498  2
; 3234          4499  2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 3235          4500  2
; 3236          4501  2     !+
; 3237          4502  2     ! Signal if the FIND failed.
; 3238          4503  2     !-
; 3239          4504  2     IF NOT .RMS_STATUS
; 3240          4505  2     THEN
; 3241          4506  2         BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3242          4507  2
; 3243          4508  2     RETURN;
; 3244          4509  1     END;                                        ! End of BAS$$REC_FRE


                    1E  AB          01  90 00000 BAS$$REC_FRE::
                                                 MOVB    #1, 30(CCB)                          ; 4480
                    04  AB      04  AE  C8 00004         BISL2   LOCK_FLAGS, 4(CCB)           ; 4486
                                    5B  DD 00009         PUSHL   CCB                          ; 4492
            00000000G   00          01  FB 0000B         CALLS   #1, SYS$FIND                 ;
                    04  AB      04  AE  CC 00012         XORL2   LOCK_FLAGS, 4(CCB)           ; 4499
                        0A          50  E8 00017         BLBS    RMS_STATUS, 1$               ; 4504
                        7E          01  CE 0001A         MNEGL   #1, -(SP)                    ; 4506
            00000000G   00          01  FB 0001D         CALLS   #1, BAS$$STOP_IO             ;
                                        05 00024 1$:     RSB                                  ; 4509

; Routine Size:  37 bytes,    Routine Base:  _BAS$CODE + 0B44

; 3245          4510  1
```

```
; 3247          4511  1 GLOBAL ROUTINE BAS$$REC_FIN (                    ! FIND (indexed) a record
; 3248          4512  1         KEY_NO, REL_OP, KEY, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
; 3249          4513  1
; 3250          4514  1 !++
; 3251          4515  1 ! FUNCTIONAL DESCRIPTION:
; 3252          4516  1 !
; 3253          4517  1 !       Find indicated record.  If successful then return; otherwise, signal a fatal
; 3254          4518  1 !       error.
; 3255          4519  1 !
; 3256          4520  1 ! FORMAL PARAMETERS:
; 3257          4521  1 !
; 3258          4522  1 !       KEY_NO.rl.v                     key of reference
; 3259          4523  1 !       REL_OP.rl.v                     relational operator for key
; 3260          4524  1 !       KEY.rt.dx                       key to search for
; 3261          4525  1 !       LOCK_FLAGS.rlu.v                bits to set in the RAB ROP to control
; 3262          4526  1 !                                       manual record locking (0 if none)
; 3263          4527  1 !
; 3264          4528  1 ! IMPLICIT INPUTS:
; 3265          4529  1 !
; 3266          4530  1 !       NONE
; 3267          4531  1 !
; 3268          4532  1 ! IMPLICIT OUTPUTS:
; 3269          4533  1 !
; 3270          4534  1 !       RAB$L_KBF               pointer to the desired key value
; 3271          4535  1 !       RAB$B_KSZ               size of desired key value
; 3272          4536  1 !       RAB$M_KGE               relational in RAB$L_ROP indicating greater than
; 3273          4537  1 !                               or equal
; 3274          4538  1 !       RAB$M_KGT               relational in RAB$L_ROP indicating greater than
; 3275          4539  1 !       RAB$B_KRF               indicates key of reference
; 3276          4540  1 !       RAB$B_RAC               record access field
; 3277          4541  1 !
; 3278          4542  1 ! ROUTINE VALUE:
; 3279          4543  1 !
; 3280          4544  1 !       NONE
; 3281          4545  1 !
; 3282          4546  1 ! SIDE EFFECTS:
; 3283          4547  1 !
; 3284          4548  1 !       See RMS Reference manual for discussion on whether match will be exact,
; 3285          4549  1 !       generic, approximate, or generic-approximate.
; 3286          4550  1 !       SIGNALs any RMS errors
; 3287          4551  1 !--
; 3288          4552  1
; 3289          4553  2     BEGIN
; 3290          4554  2
; 3291          4555  2     EXTERNAL REGISTER
; 3292          4556  2         CCB : REF BLOCK [, BYTE];
; 3293          4557  2
; 3294          4558  2     MAP
; 3295          4559  2         KEY : REF BLOCK [8, BYTE];
; 3296          4560  2
; 3297          4561  2     LITERAL
; 3298          4562  2         K_EQUAL = 0,                            ! search for key equal
; 3299          4563  2         K_GREATER_EQUAL = 1,                    ! search for key GEQ
; 3300          4564  2         K_GREATER_THAN = 2;                     ! search for key GTR
; 3301          4565  2
; 3302          4566  2     LOCAL
; 3303          4567  2         RMS_STATUS;
```

```
; 3304          4568  2
; 3305          4569  2      !+
; 3306          4570  2      ! Set the key buffer field, the key size field, the key of reference,
; 3307          4571  2      ! and the relational bits in the ROP.
; 3308          4572  2      ! Set the record access field in the RAB to key.  Perform the FIND.
; 3309          4573  2      ! If RMS returns a failure status, signal the error.
; 3310          4574  2      !-
; 3311          4575  2
; 3312          4576  2      CCB [RAB$B_RAC] = RAB$C_KEY;
; 3313          4577  2      CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];
; 3314          4578  2      CCB [RAB$B_KRF] = .KEY_NO;
; 3315          4579  3      CCB [RAB$B_KSZ] = (IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
; 3316          4580  3                         THEN
; 3317          4581  3                              .KEY [DSC$W_LENGTH]
; 3318          4582  3                         ELSE
; 3319          4583  2                              (.KEY [DSC$W_LENGTH]/2 + 1));
; 3320          4584  2
; 3321          4585  2      CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF
; 3322          4586  2          SET
; 3323          4587  2
; 3324          4588  2          [K_EQUAL] :
; 3325          4589  2              CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;
; 3326          4590  2
; 3327          4591  2          [K_GREATER_EQUAL] :
; 3328          4592  3              BEGIN
; 3329          4593  3              CCB [RAB$V_KGE] = 1;
; 3330          4594  3              CCB [RAB$V_KGT] = 0;
; 3331          4595  2              END;
; 3332          4596  2
; 3333          4597  2          [K_GREATER_THAN] :
; 3334          4598  3              BEGIN
; 3335          4599  3              CCB [RAB$V_KGT] = 1;
; 3336          4600  3              CCB [RAB$V_KGE] = 0;
; 3337          4601  2              END;
; 3338          4602  2          TES;
; 3339          4603  2
; 3340          4604  2      !+
; 3341          4605  2      ! Set bits in the RAB ROP without clearing ULK.
; 3342          4606  2      !-
; 3343          4607  2
; 3344          4608  2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
; 3345          4609  2
; 3346          4610  2      !+
; 3347          4611  2      ! perform the FIND.
; 3348          4612  2      !-
; 3349          4613  2
; 3350          4614  2      RMS_STATUS = $FIND (RAB = .CCB);
; 3351          4615  2
; 3352          4616  2      !+
; 3353          4617  2      ! Turn off bits in the RAB ROP so that subsequent operations do not
; 3354          4618  2      ! inherit them.
; 3355          4619  2      !-
; 3356          4620  2
; 3357          4621  2      CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
; 3358          4622  2
; 3359          4623  2      !+
; 3360          4624  2      ! Signal if the FIND failed.
```

```
; 3361          4625  2     !-
; 3362          4626  2     IF NOT .RMS_STATUS
; 3363          4627  2     THEN
; 3364          4628  2         BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3365          4629  2
; 3366          4630  2     RETURN;
; 3367          4631  1     END;                                    ! End of BAS$$REC_FIN


                                    52  DD 00000 BAS$$REC_FIN::
                                                    PUSHL   R2                          ; 4511
                        1E  AB          01  90 00002        MOVB    #1, 30(CCB)         ; 4576
                        30  AB      04  A2  D0 00006        MOVL    4(KEY), 48(CCB)     ; 4577
                        35  AB          50  90 0000B        MOVB    KEY_NO, 53(CCB)     ; 4578
                            15      02  A2  91 0000F        CMPB    2(KEY), #21         ; 4579
                                        05  13 00013        BEQL    1$                  ;
                            52          62  3C 00015        MOVZWL  (KEY), R2           ; 4581
                                        08  11 00018        BRB     2$                  ;
                            52          62  3C 0001A 1$:    MOVZWL  (KEY), R2           ; 4583
                            52          02  C6 0001D        DIVL2   #2, R2              ;
                                        52  D6 00020        INCL    R2                  ;
                        34  AB          52  90 00022 2$:    MOVB    R2, 52(CCB)         ; 4579
                            52      04  AB  9E 00026        MOVAB   4(CCB), R2          ; 4589
                    02      00          51  CF 0002A        CASEL   REL_OP, #0, #2      ; 4585
                  0018    000D        0006     0002E 3$:    .WORD   4$-3$,-             ;
                                                             5$-3$,-                     ;
                                                             6$-3$                       ;
                        02  A2      40  8F  8A 00034 4$:    BICB2   #64, 2(R2)          ; 4589
                                        10  11 00039        BRB     7$                  ;
                        02  A2          20  88 0003B 5$:    BISB2   #32, 2(R2)          ; 4593
                        02  A2      40  8F  8A 0003F        BICB2   #64, 2(R2)          ; 4594
                                        09  11 00044        BRB     8$                  ; 4585
                        02  A2      40  8F  88 00046 6$:    BISB2   #64, 2(R2)          ; 4599
                        02  A2          20  8A 0004B 7$:    BICB2   #32, 2(R2)          ; 4600
                            62          53  C8 0004F 8$:    BISL2   LOCK_FLAGS, (R2)    ; 4608
                                        5B  DD 00052        PUSHL   CCB                 ; 4614
                00000000G   00          01  FB 00054        CALLS   #1, SYS$FIND        ;
                            62          53  CC 0005B        XORL2   LOCK_FLAGS, (R2)    ; 4621
                            0A          50  E8 0005E        BLBS    RMS_STATUS, 9$      ; 4626
                            7E          01  CE 00061        MNEGL   #1, -(SP)           ; 4628
                00000000G   00          01  FB 00064        CALLS   #1, BAS$$STOP_IO    ;
                                        04  BA 0006B 9$:    POPR    #^M<R2>             ; 4631
                                            05 0006D        RSB                         ;

; Routine Size:  110 bytes,    Routine Base:  _BAS$CODE + 0B69

; 3368          4632  1
```

```
; 3370          4633  1 GLOBAL ROUTINE BAS$$REC_DSE                    . DELETE (sequential) a record
; 3371          4634  1     : JSB_REC0 NOVALUE =
; 3372          4635  1
; 3373          4636  1 !+
; 3374          4637  1 ! FUNCTIONAL DESCRIPTION:
; 3375          4638  1 !
; 3376          4639  1 !       Delete current record.  If successful then return; otherwise, signal a fatal
; 3377          4640  1 !       error.
; 3378          4641  1 !
; 3379          4642  1 ! FORMAL PARAMETERS:
; 3380          4643  1 !
; 3381          4644  1 !       NONE
; 3382          4645  1 !
; 3383          4646  1 ! IMPLICIT INPUTS:
; 3384          4647  1 !
; 3385          4648  1 !       NONE
; 3386          4649  1 !
; 3387          4650  1 ! IMPLICIT OUTPUTS:
; 3388          4651  1 !
; 3389          4652  1 !       RAB$B_RAC               record access field
; 3390          4653  1 !
; 3391          4654  1 ! ROUTINE VALUE:
; 3392          4655  1 !
; 3393          4656  1 !       NONE
; 3394          4657  1 !
; 3395          4658  1 ! SIDE EFFECTS:
; 3396          4659  1 !
; 3397          4660  1 !       SIGNALs any RMS errors
; 3398          4661  1 !--
; 3399          4662  1
; 3400          4663  2     BEGIN
; 3401          4664  2
; 3402          4665  2     EXTERNAL REGISTER
; 3403          4666  2         CCB : REF BLOCK [, BYTE];
; 3404          4667  2
; 3405          4668  2     !+
; 3406          4669  2     ! Set the record access field in the RAB to sequential.  Perform the DELETE.
; 3407          4670  2     ! If RMS returns a failure status, signal the error.
; 3408          4671  2     !-
; 3409          4672  2
; 3410          4673  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3411          4674  2
; 3412          4675  2     IF NOT $DELETE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3413          4676  2
; 3414          4677  2     RETURN;
; 3415          4678  1     END;                                        ! End of BAS$$REC_DSE


                                                        .EXTRN  SYS$DELETE

                                1E   AB  04 00000 BAS$$REC_DSE::
                                                        CLRB    30(CCB)                                 ; 4673
                                     5B  DD 00003       PUSHL   CCB                                     ; 4675
                00000000G  00        01  FB 00005       CALLS   #1, SYS$DELETE                          ;
                           0A        50  E8 0000C       BLBS    R0, 1$                                  ;
                           7E        01  CE 0000F       MNEGL   #1, -(SP)                               ;
```

```
                    00000000G  00          01  FB 00012          CALLS   #1, BAS$$STOP_IO                    ;
                                               05 00019 1$.       RSB                                       ; 4678

; Routine Size:  26 bytes,    Routine Base:  _BAS$CODE + 0BD7

; 3416          4679  1
```

```
: 3418          4680  1 GLOBAL ROUTINE BAS$$REC_UNL                    ! UNLOCK a record
: 3419          4681  1     : JSB_REC0 NOVALUE =
: 3420          4682  1
: 3421          4683  1 !++
: 3422          4684  1 ! FUNCTIONAL DESCRIPTION:
: 3423          4685  1 !
: 3424          4686  1 !       Unlock the current record.  If successful or no records locked,
: 3425          4687  1 !       then return; otherwise, signal a fatal error.
: 3426          4688  1 !
: 3427          4689  1 ! FORMAL PARAMETERS:
: 3428          4690  1 !
: 3429          4691  1 !       NONE
: 3430          4692  1 !
: 3431          4693  1 ! IMPLICIT INPUTS:
: 3432          4694  1 !
: 3433          4695  1 !       NONE
: 3434          4696  1 !
: 3435          4697  1 ! IMPLICIT OUTPUTS:
: 3436          4698  1 !
: 3437          4699  1 !       RAB$B_RAC               record access field
: 3438          4700  1 !
: 3439          4701  1 ! ROUTINE VALUE:
: 3440          4702  1 !
: 3441          4703  1 !       NONE
: 3442          4704  1 !
: 3443          4705  1 ! SIDE EFFECTS:
: 3444          4706  1 !
: 3445          4707  1 !       SIGNALs any RMS errors
: 3446          4708  1 !--
: 3447          4709  1
: 3448          4710  2     BEGIN
: 3449          4711  2
: 3450          4712  2     EXTERNAL REGISTER
: 3451          4713  2         CCB : REF BLOCK [, BYTE];
: 3452          4714  2
: 3453          4715  2     !+
: 3454          4716  2     ! Set the record access field in the RAB to sequential.  Perform the UNLOCK.
: 3455          4717  2     ! If RMS returns a failure status, signal the error.
: 3456          4718  2     !-
: 3457          4719  2
: 3458          4720  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3459          4721  2
: 3460          4722  3     IF NOT $RELEASE (RAB = .CCB)
: 3461          4723  2     THEN
: 3462          4724  2
: 3463          4725  2         IF .CCB [RAB$L_STS] NEQ RMS$_RNL
: 3464          4726  2         THEN
: 3465          4727  2 !+
: 3466          4728  2 ! An error was returned, check for 'record not locked'.
: 3467          4729  2 !-
: 3468          4730  2             BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3469          4731  2
: 3470          4732  2     RETURN;
: 3471          4733  1     END;                                        ! End of BAS$$REC_UNL
```

```
                                                        .EXTRN  SYS$RELEASE

                                1E   AB  94 00000 BAS$$REC_UNL::
                                                        CLRB    30(CCB)                          ; 4720
                                     5B  DD 00003       PUSHL   CCB                              ; 4722
                00000000G  00        01  FB 00005       CALLS   #1, SYS$RELEASE                  ;
                           14        50  E8 0000C       BLBS    R0, 1$                           ;
                000181A0   8F   08   AB  D1 0000F       CMPL    8(CCB), #98720                   ; 4725
                                     0A  13 00017       BEQL    1$                               ;
                           7E        01  CE 00019       MNEGL   #1, -(SP)                        ; 4730
                00000000G  00        01  FB 0001C       CALLS   #1, BAS$$STOP_IO                 ;
                                         05 00023 1$:   RSB                                      ; 4733

; Routine Size:  36 bytes,    Routine Base:  _BAS$CODE + 0BF1

; 3472          4734  1
```

```
; 3474          4735  1 GLOBAL ROUTINE BAS$$REC_FEE                    ! FREE all locked records
; 3475          4736  1     : JSB_REC0 NOVALUE =
; 3476          4737  1
; 3477          4738  1 !++
; 3478          4739  1 ! FUNCTIONAL DESCRIPTION:
; 3479          4740  1 !
; 3480          4741  1 !       Free all locked records.  If successful or no records locked,
; 3481          4742  1 !       then return; otherwise, signal a fatal error.
; 3482          4743  1 !
; 3483          4744  1 ! FORMAL PARAMETERS:
; 3484          4745  1 !
; 3485          4746  1 !       NONE
; 3486          4747  1 !
; 3487          4748  1 ! IMPLICIT INPUTS:
; 3488          4749  1 !
; 3489          4750  1 !       NONE
; 3490          4751  1 !
; 3491          4752  1 ! IMPLICIT OUTPUTS:
; 3492          4753  1 !
; 3493          4754  1 !       RAB$B_RAC               record access field
; 3494          4755  1 !
; 3495          4756  1 ! ROUTINE VALUE:
; 3496          4757  1 !
; 3497          4758  1 !       NONE
; 3498          4759  1 !
; 3499          4760  1 ! SIDE EFFECTS:
; 3500          4761  1 !
; 3501          4762  1 !       SIGNALs any RMS errors
; 3502          4763  1 !--
; 3503          4764  1
; 3504          4765  2     BEGIN
; 3505          4766  2
; 3506          4767  2     EXTERNAL REGISTER
; 3507          4768  2         CCB : REF BLOCK [, BYTE];
; 3508          4769  2
; 3509          4770  2     !+
; 3510          4771  2     ! Set the record access field in the RAB to sequential.  Perform the FREE.
; 3511          4772  2     ! If RMS returns a failure status, signal the error.
; 3512          4773  2     !-
; 3513          4774  2
; 3514          4775  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3515          4776  2
; 3516          4777  3     IF NOT $FREE (RAB = .CCB)
; 3517          4778  2     THEN
; 3518          4779  2
; 3519          4780  2         IF .CCB [RAB$L_STS] NEQ RMS$_RNL
; 3520          4781  2         THEN
; 3521          4782  2 !+
; 3522          4783  2 ! An error was returned, check for "record not locked".
; 3523          4784  2 !-
; 3524          4785  2             BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3525          4786  2
; 3526          4787  2     RETURN;
; 3527          4788  1     END;                                    ! End of BAS$$REC_FEE
```

```
                                                        .EXTRN  SYS$FREE

                                 1E   AB  94 00000 BAS$$REC_FEE::
                                                        CLRB    30(CCB)                                  ; 4775
                                      5B  DD 00003      PUSHL   CCB                                      ; 4777
                 00000000G  00        01  FB 00005      CALLS   #1, SYS$FREE                             ;
                            14        50  E8 0000C      BLBS    R0, 1$                                   ;
                 000181A0   8F    08  AB  D1 0000F      CMPL    8(CCB), #98720                           ; 4780
                                      0A  13 00017      BEQL    1$                                       ;
                            7E        01  CE 00019      MNEGL   #1, -(SP)                                ; 4785
                 00000000G  00        01  FB 0001C      CALLS   #1, BAS$$STOP_IO                         ;
                                          05 00023 1$:  RSB                                              ; 4788

; Routine Size:  36 bytes,    Routine Base:  _BAS$CODE + 0C15

; 3528          4789  1
```

```
; 3530   4790  1 GLOBAL ROUTINE BAS$$REC_UPD (                    ! UPDATE a record
; 3531   4791  1         COUNT                                    ! No. of bytes in the record
; 3532   4792  1     ) : JSB_DO_WRITE NOVALUE =
; 3533   4793  1
; 3534   4794  1 !++
; 3535   4795  1 ! FUNCTIONAL DESCRIPTION:
; 3536   4796  1 !
; 3537   4797  1 !       Update current record.  If successful then return; otherwise, signal a fatal
; 3538   4798  1 !       error.
; 3539   4799  1 !
; 3540   4800  1 ! FORMAL PARAMETERS:
; 3541   4801  1 !
; 3542   4802  1 !       COUNT.rl.v              No. of bytes in record to update
; 3543   4803  1 !
; 3544   4804  1 ! IMPLICIT INPUTS:
; 3545   4805  1 !
; 3546   4806  1 !       NONE
; 3547   4807  1 !
; 3548   4808  1 ! IMPLICIT OUTPUTS:
; 3549   4809  1 !
; 3550   4810  1 !       RAB$B_RAC               record access field
; 3551   4811  1 !       RAB$W_RSZ               record size
; 3552   4812  1 !
; 3553   4813  1 ! ROUTINE VALUE:
; 3554   4814  1 !
; 3555   4815  1 !       NONE
; 3556   4816  1 !
; 3557   4817  1 ! SIDE EFFECTS:
; 3558   4818  1 !
; 3559   4819  1 !       Update current record in file on this logical unit.
; 3560   4820  1 !       SIGNALs any RMS errors
; 3561   4821  1 !--
; 3562   4822  1
; 3563   4823  2     BEGIN
; 3564   4824  2
; 3565   4825  2     EXTERNAL REGISTER
; 3566   4826  2         CCB : REF BLOCK [, BYTE];
; 3567   4827  2
; 3568   4828  2 !+
; 3569   4829  2 ! Point RBF to the user buffer.
; 3570   4830  2 ! Set the record access field in the RAB to sequential.  Perform the UPDATE.
; 3571   4831  2 ! If RMS returns a failure status, signal the error.
; 3572   4832  2 !-
; 3573   4833  2     CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
; 3574   4834  2     CCB [RAB$W_RSZ] = .COUNT;
; 3575   4835  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3576   4836  2
; 3577   4837  2     IF NOT $UPDATE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_ICERR_REC);
; 3578   4838  2
; 3579   4839  2 !+
; 3580   4840  2 ! Point LUB$A_RBUF_ADR to the buffer used by RMS for MOVE.
; 3581   4841  2 !-
; 3582   4842  2     CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
; 3583   4843  2     RETURN;
; 3584   4844  1     END;                                         ! End of BAS$$REC_UPD
```

```
                                                        .EXTRN  SYS$UPDATE

                28   AB      24   AB  D0 00000 BAS$$REC_UPD::
                                              MOVL    36(CCB), 40(CCB)                    ; 4833
                22   AB           5C  B0 00005  MOVW    COUNT, 34(CCB)                      ; 4834
                             1E   AB  94 00009  CLRB    30(CCB)                             ; 4835
                                  5B  DD 0000C  PUSHL   CCB                                 ; 4837
        00000000G  00             01  FB 0000E  CALLS   #1, SYS$UPDATE                      ;
                   0A             50  E8 00015  BLBS    R0, 1$                              ;
                   7E             01  CE 00018  MNEGL   #1, -(SP)                           ;
        00000000G  00             01  FB 0001B  CALLS   #1, BAS$$STOP_IO                    ;
                EC   AB      24   AB  D0 00022 1$:  MOVL    36(CCB), -20(CCB)                   ; 4842
                                      05 00027  RSB                                         ; 4844

; Routine Size:  40 bytes,    Routine Base:  _BAS$CODE + 0C39

; 3585          4845  1
```

```
; 3587          4846  1 GLOBAL ROUTINE BAS$$REC_RSE                    ! RESTORE (sequential) to beginning of file
; 3588          4847  1     : JSB_REC0 NOVALUE =
; 3589          4848  1
; 3590          4849  1 !++
; 3591          4850  1 ! FUNCTIONAL DESCRIPTION:
; 3592          4851  1 !
; 3593          4852  1 !       Rewind the file.  If successful then return; otherwise, signal a fatal
; 3594          4853  1 !       error.
; 3595          4854  1 !
; 3596          4855  1 ! FORMAL PARAMETERS:
; 3597          4856  1 !
; 3598          4857  1 !       NONE
; 3599          4858  1 !
; 3600          4859  1 ! IMPLICIT INPUTS:
; 3601          4860  1 !
; 3602          4861  1 !       NONE
; 3603          4862  1 !
; 3604          4863  1 ! IMPLICIT OUTPUTS:
; 3605          4864  1 !
; 3606          4865  1 !       RAB$B_RAC               record access field
; 3607          4866  1 !
; 3608          4867  1 ! ROUTINE VALUE:
; 3609          4868  1 !
; 3610          4869  1 !       NONE
; 3611          4870  1 !
; 3612          4871  1 ! SIDE EFFECTS:
; 3613          4872  1 !
; 3614          4873  1 !       SIGNALs any RMS errors
; 3615          4874  1 !--
; 3616          4875  1
; 3617          4876  2     BEGIN
; 3618          4877  2
; 3619          4878  2     EXTERNAL REGISTER
; 3620          4879  2         CCB : REF BLOCK [, BYTE];
; 3621          4880  2
; 3622          4881  2     !+
; 3623          4882  2     ! Set the record access field in the RAB to sequential.  Perform the REWIND.
; 3624          4883  2     ! If RMS returns a failure status, signal the error.
; 3625          4884  2     !-
; 3626          4885  2
; 3627          4886  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3628          4887  2
; 3629          4888  2     IF NOT $REWIND (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3630          4889  2
; 3631          4890  2     RETURN;
; 3632          4891  1     END;                                        ! End of BAS$$REC_RSE


                                                        .EXTRN  SYS$REWIND

                                1E   AB  94 00000 BAS$$REC_RSE::
                                                        CLRB    30(CCB)                                     ; 4886
                                     5B  DD 00003       PUSHL   CCB                                         ; 4888
                00000000G  00        01  FB 00005       CALLS   #1, SYS$REWIND                              ;
                           0A        50  E8 0000C       BLBS    R0, 1$                                      ;
                           7E        01  CE 0000F       MNEGL   #1, -(SP)                                   ;
```

```
                    00000000G  00          01  FB 00012          CALLS   #1, BAS$$STOP_IO                    ;
                                               05 00019 1$:      RSB                                         ; 4891

; Routine Size:  26 bytes,    Routine Base:  _BAS$CODE + 0C61

; 3633          4892  1
```

```
; 3635          4893  1 GLOBAL ROUTINE BAS$$REC_RIN (                    ! RESTORE (indexed) to beginning of file
; 3636          4894  1         KEY_NO) : JSB_REC_IND NOVALUE =
; 3637          4895  1
; 3638          4896  1 !++
; 3639          4897  1 ! FUNCTIONAL DESCRIPTION:
; 3640          4898  1 !
; 3641          4899  1 !       Rewind the file.  If successful then return; otherwise, signal a fatal
; 3642          4900  1 !       error.
; 3643          4901  1 !
; 3644          4902  1 ! FORMAL PARAMETERS:
; 3645          4903  1 !
; 3646          4904  1 !       KEY_NO.rl.v                     key of reference
; 3647          4905  1 !
; 3648          4906  1 ! IMPLICIT INPUTS:
; 3649          4907  1 !
; 3650          4908  1 !       NONE
; 3651          4909  1 !
; 3652          4910  1 ! IMPLICIT OUTPUTS:
; 3653          4911  1 !
; 3654          4912  1 !       RAB$B_KRF               key of reference
; 3655          4913  1 !       RAB$B_RAC               record access field
; 3656          4914  1 !
; 3657          4915  1 ! ROUTINE VALUE:
; 3658          4916  1 !
; 3659          4917  1 !       NONE
; 3660          4918  1 !
; 3661          4919  1 ! SIDE EFFECTS:
; 3662          4920  1 !
; 3663          4921  1 !       SIGNALs any RMS errors
; 3664          4922  1 !--
; 3665          4923  1
; 3666          4924  2     BEGIN
; 3667          4925  2
; 3668          4926  2     EXTERNAL REGISTER
; 3669          4927  2         CCB : REF BLOCK [, BYTE];
; 3670          4928  2
; 3671          4929  2     !+
; 3672          4930  2     ! Set the key of reference .
; 3673          4931  2     ! Set the record access field in the RAB to key.  Perform the REWIND.
; 3674          4932  2     ! If RMS returns a failure status, signal the error.
; 3675          4933  2     !-
; 3676          4934  2
; 3677          4935  2     CCB [RAB$B_KRF] = .KEY_NO;
; 3678          4936  2     CCB [RAB$B_RAC] = RAB$C_KEY;
; 3679          4937  2
; 3680          4938  2     IF NOT $REWIND (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3681          4939  2
; 3682          4940  2     RETURN;
; 3683          4941  1     END;                                        ! End of BAS$$REC_RIN


                35    AB            50  90 00000 BAS$$REC_RIN::
                                                 MOVB    KEY_NO, 53(CCB)                         ; 4935
                1E    AB            01  90 00004         MOVB    #1, 30(CCB)                     ; 4936
```

```
                                  5B  DD 00008          PUSHL   CCB                                  ; 4938
00000000G  00                     01  FB 0000A          CALLS   #1, SYS$REWIND                       ;
           0A                     50  E8 00011          BLBS    R0, 1$                               ;
           7E                     01  CE 00014          MNEGL   #1, -(SP)                            ;
00000000G  00                     01  FB 00017          CALLS   #1, BAS$$STOP_IO                     ;
                                      05 0001E 1$:       RSB                                          ; 4941

; Routine Size:  31 bytes,    Routine Base:  _BAS$CODE + 0C7B

; 3684          4942  1
```

```
; 3686          4943  1 GLOBAL ROUTINE BAS$$REC_SSE                    ! SCRATCH (sequential) a record
; 3687          4944  1     : JSB_RECO NOVALUE =
; 3688          4945  1
; 3689          4946  1 !++
; 3690          4947  1 ! FUNCTIONAL DESCRIPTION:
; 3691          4948  1 !
; 3692          4949  1 !       Truncate this file.  If successful then return; otherwise, signal a fatal
; 3693          4950  1 !       error.
; 3694          4951  1 !
; 3695          4952  1 ! FORMAL PARAMETERS:
; 3696          4953  1 !
; 3697          4954  1 !       NONE
; 3698          4955  1 !
; 3699          4956  1 ! IMPLICIT INPUTS:
; 3700          4957  1 !
; 3701          4958  1 !       NONE
; 3702          4959  1 !
; 3703          4960  1 ! IMPLICIT OUTPUTS:
; 3704          4961  1 !
; 3705          4962  1 !       RAB$B_RAC               record access field
; 3706          4963  1 !
; 3707          4964  1 ! ROUTINE VALUE:
; 3708          4965  1 !
; 3709          4966  1 !       NONE
; 3710          4967  1 !
; 3711          4968  1 ! SIDE EFFECTS:
; 3712          4969  1 !
; 3713          4970  1 !       SIGNALs any RMS errors
; 3714          4971  1 !--
; 3715          4972  1
; 3716          4973  2     BEGIN
; 3717          4974  2
; 3718          4975  2     EXTERNAL REGISTER
; 3719          4976  2         CCB : REF BLOCK [, BYTE];
; 3720          4977  2
; 3721          4978  2     !+
; 3722          4979  2     ! Set the record access field in the RAB to sequential.  Perform the TRUNCATE.
; 3723          4980  2     ! If RMS returns a failure status, signal the error.
; 3724          4981  2     !-
; 3725          4982  2
; 3726          4983  2     CCB [RAB$B_RAC] = RAB$C_SEQ;
; 3727          4984  2
; 3728          4985  2     IF NOT $TRUNCATE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3729          4986  2
; 3730          4987  2     RETURN;
; 3731          4988  1     END;                                        ! End of BAS$$REC_SSE


                                                        .EXTRN  SYS$TRUNCATE

                                1E   AB  94 00000 BAS$$REC_SSE::
                                                        CLRB    30(CCB)                                 ; 4983
                                     5B  DD 00003       PUSHL   CCB                                     ; 4985
                00000000G  00        01  FB 00005       CALLS   #1, SYS$TRUNCATE                        ;
                           0A        50  E8 0000C       BLBS    R0, 1$                                  ;
                           7E        01  CE 0000F       MNEGL   #1, -(SP)                               ;
```

```
                    00000000G  00          01  FB 00012          CALLS   #1, BAS$$STOP_IO                    ;
                                               05 00019 1$:      RSB                                        ; 4988

; Routine Size:  26 bytes,    Routine Base:  _BAS$CODE + 0C9A

; 3732          4989  1
```

```
; 3734          4990  1 ROUTINE PUT_ERROR (                          ! Here on error in $PUT
; 3735          4991  1         SIGNAL_OR_STOP                       ! specifies whether to signal or stop
; 3736          4992  1     ) : CALL_CCB NOVALUE =
; 3737          4993  1
; 3738          4994  1 !++
; 3739          4995  1 ! FUNCTIONAL DESCRIPTION:
; 3740          4996  1 !
; 3741          4997  1 !       Here on $PUT errors, check for Record stream active error (RMS$_RSA)
; 3742          4998  1 !       If this error, WAIT until not active and try $PUT again.
; 3743          4999  1 !       This recovers from AST I/O which can occur out of the middle
; 3744          5000  1 !       of synchronous I/O at non-AST level.
; 3745          5001  1 !
; 3746          5002  1 ! CALLING SEQUENCE:
; 3747          5003  1 !
; 3748          5004  1 !       PUT_ERROR (signal_or_stop)
; 3749          5005  1 !
; 3750          5006  1 ! FORMAL PARAMETERS:
; 3751          5007  1 !
; 3752          5008  1 !       SIGNAL_OR_STOP.rl.v             whether to signal or stop
; 3753          5009  1 !
; 3754          5010  1 ! IMPLICIT INPUTS:
; 3755          5011  1 !
; 3756          5012  1 !       CCB                     Adr. of current LUB/ISB/RAB
; 3757          5013  1 !
; 3758          5014  1 ! IMPLICIT OUTPUTS:
; 3759          5015  1 !
; 3760          5016  1 !       LUB$V_OUTBUF_DR         Cleared to indicate clean buffer
; 3761          5017  1 !
; 3762          5018  1 ! ROUTINE VALUE:
; 3763          5019  1 !
; 3764          5020  1 !       NONE
; 3765          5021  1 !
; 3766          5022  1 ! SIDE EFFECTS:
; 3767          5023  1 !
; 3768          5024  1 !       $WAITs and then tries $PUT again, until success or any error
; 3769          5025  1 !       except record streanm active.
; 3770          5026  1 !--
; 3771          5027  1
; 3772          5028  2     BEGIN
; 3773          5029  2
; 3774          5030  2     EXTERNAL REGISTER
; 3775          5031  2         CCB : REF BLOCK [, BYTE];
; 3776          5032  2
; 3777          5033  2     WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
; 3778          5034  3         BEGIN
; 3779          5035  3         $WAIT (RAB = .CCB);
; 3780          5036  4         $PUT (RAB = .CCB)
; 3781          5037  2         END;
; 3782          5038  2
; 3783          5039  2     IF NOT .CCB [RAB$L_STS]
; 3784          5040  2     THEN
; 3785          5041  3         BEGIN
; 3786          5042  3 !+
; 3787          5043  3 ! Clear the buffer dirty bit so if there is anything there BAS$CLOSE won't
; 3788          5044  3 ! get confused, and try to do another PUT.
; 3789          5045  3 !-
; 3790          5046  3         CCB [LUB$V_OUTBUF_DR] = 0;
```

```
; 3791          5047  3
; 3792          5048  3         IF .SIGNAL_OR_STOP EQL K_SIGNAL
; 3793          5049  3         THEN
; 3794          5050  3             BAS$$SIGNAL_IO (BAS$K_IOERR_REC)
; 3795          5051  3         ELSE
; 3796          5052  3             BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3797          5053  3
; 3798          5054  2         END;
; 3799          5055  2     RETURN;
; 3800          5056  1     END;                                    ! End of PUT_ERROR


                                  0000 00000 PUT_ERROR:
                                                     .WORD   Save nothing                          ; 4990
000182DA   8F        08   AB  D1 00002 1$:        CMPL    8(CCB), #99034                        ; 5033
                          14  12 0000A            BNEQ    2$
                          5B  DD 0000C            PUSHL   CCB                                   ; 5035
00000000G  00             01  FB 0000E            CALLS   #1, SYS$WAIT
                          5B  DD 00015            PUSHL   CCB                                   ; 5036
00000000G  00             01  FB 00017            CALLS   #1, SYS$PUT
                          E2  11 0001E            BRB     1$
           1F        08   AB  E8 00020 2$:        BLBS    8(CCB), 4$                            ; 5039
      FE   AB             08  8A 00024            BICB2   #8, -2(CCB)                           ; 5046
           01        04   AC  D1 00028            CMPL    SIGNAL_OR_STOP, #1                    ; 5048
                          0B  12 0002C            BNEQ    3$
           7E             01  CE 0002E            MNEGL   #1, -(SP)                             ; 5050
00000000G  00             01  FB 00031            CALLS   #1, BAS$$SIGNAL_IO
                             04 00038             RET
           7E             01  CE 00039 3$:        MNEGL   #1, -(SP)                             ; 5052
00000000G  00             01  FB 0003C            CALLS   #1, BAS$$STOP_IO
                             04 00043 4$:         RET                                           ; 5056

; Routine Size:  68 bytes,    Routine Base:  _BAS$CODE + 0CB4
```

```
; 3802          5057  1 ROUTINE GET_ERROR (                                  ! Here on error on $GET
; 3803          5058  1         SIGNAL_OR_STOP                               ! parameter to signal(continue) or stop
; 3804          5059  1     ) : CALL_CCB NOVALUE =
; 3805          5060  1
; 3806          5061  1 !++
; 3807          5062  1 ! FUNCTIONAL DESCRIPTION:
; 3808          5063  1 !
; 3809          5064  1 !       Here on $GET errors, check for Record stream active error (RMS$_RSA)
; 3810          5065  1 !       If this error, WAIT until not active and try $GET again.
; 3811          5066  1 !       This recovers from AST I/O which can occur out of the middle
; 3812          5067  1 !       of synchronous I/O at non-AST level.
; 3813          5068  1 !
; 3814          5069  1 ! CALLING SEQUENCE:
; 3815          5070  1 !
; 3816          5071  1 !       JSB GET_ERROR ()
; 3817          5072  1 !
; 3818          5073  1 ! FORMAL PARAMETERS:
; 3819          5074  1 !
; 3820          5075  1 !       NONE
; 3821          5076  1 !
; 3822          5077  1 ! IMPLICIT INPUTS:
; 3823          5078  1 !
; 3824          5079  1 !       CCB                     Adr. of current LUB/ISB/RAB
; 3825          5080  1 !
; 3826          5081  1 ! IMPLICIT OUTPUTS:
; 3827          5082  1 !
; 3828          5083  1 ! ROUTINE VALUE:
; 3829          5084  1 !
; 3830          5085  1 !       NONE
; 3831          5086  1 !
; 3832          5087  1 ! SIDE EFFECTS:
; 3833          5088  1 !
; 3834          5089  1 !       If this is an INPUT LINE and a ^Z error, then just return and it will
; 3835          5090  1 !       be handled above.
; 3836          5091  1 !       $WAITs and then tries $GET again, until success or any error
; 3837          5092  1 !       except record streanm active.
; 3838          5093  1 !--
; 3839          5094  1
; 3840          5095  2     BEGIN
; 3841          5096  2
; 3842          5097  2     EXTERNAL REGISTER
; 3843          5098  2         CCB : REF BLOCK [, BYTE];
; 3844          5099  2
; 3845          5100  2 !+
; 3846          5101  2 ! Set the prompt buffer length to zero so that error followed by RESUME will not
; 3847          5102  2 ! keep concatenating the prompt buffer.
; 3848          5103  2 !-
; 3849          5104  2     CCB [RAB$B_PSZ] = 0;
; 3850          5105  2 !+
; 3851          5106  2 ! Is this INPUT LINE and only a ^Z in the record?
; 3852          5107  2 !-
; 3853          5108  2
; 3854          5109  2     IF .CCB [ISB$B_STTM_TYPE] EQL ISB$K_ST_TY_INL AND .CCB [RAB$W_RSZ] EQLU 1 AND .(.CCB [RAB$L_RBF])<0, 8>
; 3855          5110  2         EQLU BAS$K_CONTROL_Z
; 3856          5111  2     THEN
; 3857          5112  2         RETURN;
; 3858          5113  2
```

```
; 3859          5114  2     WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
; 3860          5115  3         BEGIN
; 3861          5116  3         $WAIT (RAB = .CCB);
; 3862          5117  4         $GET (RAB = .CCB)
; 3863          5118  2         END;
; 3864          5119  2
; 3865          5120  2     IF NOT .CCB [RAB$L_STS]
; 3866          5121  2     THEN
; 3867          5122  2
; 3868          5123  2     !+
; 3869          5124  2     ! Check the input parameter to see if we should signal or stop.
; 3870          5125  2     !-
; 3871          5126  2
; 3872          5127  2         IF .SIGNAL_OR_STOP EQL K_SIGNAL
; 3873          5128  2         THEN
; 3874          5129  2             BAS$$SIGNAL_IO (BAS$K_IOERR_REC)
; 3875          5130  2         ELSE
; 3876          5131  2             BAS$$STOP_IO (BAS$K_IOERR_REC);
; 3877          5132  2
; 3878          5133  2     RETURN;
; 3879          5134  1     END;                                        ! End of GET_ERROR


                                    0000 00000 GET_ERROR:
                                                       .WORD   Save nothing                          ; 5057
                                 34  AB  94 00002      CLRB    52(CCB)                               ; 5104
                        20     FF71  CB  91 00005      CMPB    -143(CCB), #32                        ; 5109
                                     0C  12 0000A      BNEQ    1$                                    ;
                        01       22  AB  B1 0000C      CMPW    34(CCB), #1                           ;
                                     06  12 00010      BNEQ    1$                                    ;
                        1A       28  BB  91 00012      CMPB    @40(CCB), #26                         ; 5110
                                     3D  13 00016      BEQL    4$                                    ;
                000182DA 8F      08  AB  D1 00018 1$:  CMPL    8(CCB), #99034                        ; 5114
                                     14  12 00020      BNEQ    2$                                    ;
                                     5B  DD 00022      PUSHL   CCB                                   ; 5116
                00000000G 00         01  FB 00024      CALLS   #1, SYS$WAIT                          ;
                                     5B  DD 0002B      PUSHL   CCB                                   ; 5117
                00000000G 00         01  FB 0002D      CALLS   #1, SYS$GET                           ;
                                     E2  11 00034      BRB     1$                                    ;
                        1B       08  AB  E8 00036 2$:  BLBS    8(CCB), 4$                            ; 5120
                        01       04  AC  D1 0003A      CMPL    SIGNAL_OR_STOP, #1                    ; 5127
                                     0B  12 0003E      BNEQ    3$                                    ;
                        7E           01  CE 00040      MNEGL   #1, -(SP)                             ; 5129
                00000000G 00         01  FB 00043      CALLS   #1, BAS$$SIGNAL_IO                    ;
                                         04 0004A      RET                                           ;
                        7E           01  CE 0004B 3$:  MNEGL   #1, -(SP)                             ; 5131
                00000000G 00         01  FB 0004E      CALLS   #1, BAS$$STOP_IO                      ;
                                         04 00055 4$:  RET                                           ; 5134

; Routine Size:  86 bytes,    Routine Base:  _BAS$CODE + 0CF8

; 3880          5135  1 END
; 3881          5136  1
```

```
; 3882          5137  0 ELUDOM


                                                        BAS$$REC_WF9==      BAS$$REC_WSL9
                                                        BAS$$REC_WF1==      BAS$$REC_WSL1
                                                        BAS$$REC_WF0==      BAS$$REC_WSL0

;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _BAS$DATA                        6  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _BAS$CODE                     3406  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        44         0        581         00:01.2


;                                   COMMAND QUALIFIERS

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

; Size:          3397 code + 15 data bytes
; Run Time:         01:16.8
; Elapsed Time:     02:50.9
; Lines/CPU Min:     4014
; Lexemes/CPU-Min: 26026
; Memory Used:  249 pages
; Compilation Complete
```

BASRSTSCV
LIS

BASRAD50
LIS

BASRSET
LIS

BASPUT
LIS

BASRECPRO
LIS

BASRESTAR
LIS

BASRANDOM
LIS

BASREMAP
LIS

BASRESTOR
LIS

BASRIGHT
LIS