```
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
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    SSSSSSSS  TTTTTTTTTT    000000    RRRRRRRR
BBBBBBBB      AAAAAA      SSSSSSSS  RRRRRRRR    EEEEEEEEEE    SSSSSSSS  TTTTTTTTTT    000000    RRRRRRRR
BB      BB   AA      AA  SS        RR      RR  EE            SS              TT      00      00  RR      RR
BB      BB   AA      AA  SS        RR      RR  EE            SS              TT      00      00  RR      RR
BB      BB   AA      AA  SS        RR      RR  EE            SS              TT      00      00  RR      RR
BB      BB   AA      AA  SS        RR      RR  EE            SS              TT      00      00  RR      RR
BBBBBBBB     AA      AA    SSSSSS  RRRRRRRR    EEEEEEEE        SSSSSS        TT      00      00  RRRRRRRR
BBBBBBBB     AA      AA    SSSSSS  RRRRRRRR    EEEEEEEE        SSSSSS        TT      00      00  RRRRRRRR
BB      BB   AAAAAAAAAA          SS  RR  RR      EE                    SS      TT      00      00  RR  RR
BB      BB   AAAAAAAAAA          SS  RR  RR      EE                    SS      TT      00      00  RR  RR
BB      BB   AA      AA          SS  RR    RR    EE                    SS      TT      00      00  RR    RR
BB      BB   AA      AA          SS  RR    RR    EE                    SS      TT      00      00  RR    RR
BBBBBBBB     AA      AA  SSSSSSSS  RR      RR  EEEEEEEEEE  SSSSSSSS        TT        000000    RR      RR
BBBBBBBB     AA      AA  SSSSSSSS  RR      RR  EEEEEEEEEE  SSSSSSSS        TT        000000    RR      RR

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

```
;     1          0001  0 MODULE BAS$RESTORE (                          ! Basic RESTORE construct
;     2          0002  0                     IDENT = '1-005'           ! File: BASRESTOR.B32
;     3          0003  0                     ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:
;    32          0032  1 !       Basic support library - user callable
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !       This module is the UPI level of the Basic RESTORE construct.  Initially,
;    36          0036  1 !       it contains only the code for sequential I/O.  This module will set
;    37          0037  1 !       up the I/O data base for the LUN and go directly to the REC level.
;    38          0038  1 !       The code for RESTORE (no channel number) is appended as an
;    39          0039  1 !       afterthought.
;    40          0040  1 !
;    41          0041  1 !
;    42          0042  1 ! ENVIRONMENT:
;    43          0043  1 !       User access mode - AST reentrant.
;    44          0044  1 !
;    45          0045  1 ! AUTHOR: Donald G. Petersen, CREATION DATE: 27-Feb-79
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       DGP, 27-Feb-79 : VERSION 01
;    50          0050  1 ! 1-001 - original.  DGP 27-Feb-79
;    51          0051  1 ! 1-002 - Add code for RESTORE_DATA from old version of the
;    52          0052  1 !         RESTORE module.  JBS 28-FEB-1979
;    53          0053  1 ! 1-003 - Add RESTORE_KEY.  DGP 06-Apr-79
;    54          0054  1 ! 1-004 - Set up ISB$A_USER_FP.  JBS 25-JUL-1979
;    55          0055  1 ! 1-005 - Check for virtual array use of this file.  DGP 16-Oct-79
;    56          0056  1 !--
;    57          0057  1
```

```
;   58          0058  1 !<BLF/PAGE>
```

```
;    60          0059  1 !
;    61          0060  1 ! SWITCHES:
;    62          0061  1 !
;    63          0062  1
;    64          0063  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    65          0064  1
;    66          0065  1 !
;    67          0066  1 ! LINKAGES
;    68          0067  1 !
;    69          0068  1
;    70          0069  1 REQUIRE 'RTLIN:OTSLNK';                         ! Define all linkages
;    71          0498  1
;    72          0499  1 !
;    73          0500  1 ! TABLE OF CONTENTS:
;    74          0501  1 !
;    75          0502  1
;    76          0503  1 FORWARD ROUTINE
;    77          0504  1     BAS$RESTORE_KEY : NOVALUE,                  ! UPI level Indexed RESTORE
;    78          0505  1     BAS$RESTORE : NOVALUE,                      ! UPI level Sequential RESTORE
;    79          0506  1     BAS$RESTORE_DAT : NOVALUE;                  ! RESTORE (DATA statement)
;    80          0507  1
;    81          0508  1 !
;    82          0509  1 ! INCLUDE FILES:
;    83          0510  1 !
;    84          0511  1
;    85          0512  1 REQUIRE 'RTLIN:BASFRAME';                       ! BASIC stack frame definitions
;    86          0715  1
;    87          0716  1 REQUIRE 'RTLIN:BASINARG';                       ! Frame init argument list
;    88          0800  1
;    89          0801  1 REQUIRE 'RTLML:OTSISB';                         ! ISB definitions
;    90          0969  1
;    91          0970  1 REQUIRE 'RTLML:OTSLUB';                         ! LUB definitions
;    92          1110  1
;    93          1111  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Define DECLARE_PSECTS macro
;    94          1206  1
;    95          1207  1 LIBRARY 'RTLSTARLE';                            ! Starlet system macros
;    96          1208  1
;    97          1209  1 !
;    98          1210  1 ! MACROS:
;    99          1211  1 !
;   100          1212  1 !       NONE
;   101          1213  1 !
;   102          1214  1 ! EQUATED SYMBOLS:
;   103          1215  1 !       NONE
;   104          1216  1 !
;   105          1217  1 !
;   106          1218  1 ! PSECT DECLARATIONS:
;   107          1219  1 !
;   108          1220  1 DECLARE_PSECTS (BAS);
;   109          1221  1 !
;   110          1222  1 ! OWN STORAGE:
;   111          1223  1 !
;   112          1224  1 !       NONE
;   113          1225  1 !
;   114          1226  1 ! EXTERNAL REFERENCES:
;   115          1227  1 !
;   116          1228  1
```

```
;  117          1229  1 EXTERNAL ROUTINE
;  118          1230  1     BAS$$REC_RIN : JSB_REC_IND,                ! REC level processing - indexed I/O
;  119          1231  1     BAS$$REC_RSE : JSB_RECO NOVALUE,           ! REC level processing - RMS interface
;  120          1232  1                                                ! RESTORE sequential
;  121          1233  1     BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE,        ! Load register CCB
;  122          1234  1     BAS$$CB_POP : JSB_CB_POP NOVALUE,          ! Done with register CCB
;  123          1235  1     BAS$$STOP : NOVALUE,                       ! Signal fatal error
;  124          1236  1     BAS$$STOP_IO : NOVALUE,                    ! Signal fatal I/O error
;  125          1237  1     BAS$HANDLER;                               ! flags a BASIC frame
;  126          1238  1
;  127          1239  1 !+
;  128          1240  1 ! The following are the error codes used in this module.
;  129          1241  1 !-
;  130          1242  1
;  131          1243  1 EXTERNAL LITERAL
;  132          1244  1     BAS$K_PROLOSSOR : UNSIGNED (8),            ! Program lost, sorry
;  133          1245  1     BAS$K_ILLILLACC : UNSIGNED (8),            ! illegal or illogical access
;  134          1246  1     BAS$K_IO_CHANOT : UNSIGNED (8);            ! I/O channel not open
;  135          1247  1
```

```
;  137          1248  1 GLOBAL ROUTINE BAS$RESTORE (                  ! RESTORE sequential
;  138          1249  1                             UNIT              ! logical unit number
;  139          1250  1     ) : NOVALUE =
;  140          1251  1
;  141          1252  1 !++
;  142          1253  1 ! FUNCTIONAL DESCRIPTION:
;  143          1254  1 !
;  144          1255  1 !       This routine will set up the I/O data base for this LUN if necessary
;  145          1256  1 !       and then go directly to the REC level.  When control is returned to
;  146          1257  1 !       this routine, it pops the CCB off of the I/O system.  The actual inter-
;  147          1258  1 !       face to RMS is done at the REC level.  The file is rewound.
;  148          1259  1 !
;  149          1260  1 ! FORMAL PARAMETERS:
;  150          1261  1 !
;  151          1262  1 !       UNIT.rlu.v      logical unit number
;  152          1263  1 !
;  153          1264  1 ! IMPLICIT INPUTS:
;  154          1265  1 !
;  155          1266  1 !       LUB$V_VA_USE            virtual array use
;  156          1267  1 !
;  157          1268  1 ! IMPLICIT OUTPUTS:
;  158          1269  1 !
;  159          1270  1 !       ISB$B_STTM_TYPE         the statement type
;  160          1271  1 !       LUB$V_BLK_USE           non-virtual array use
;  161          1272  1 !
;  162          1273  1 ! COMPLETION CODES:
;  163          1274  1 !
;  164          1275  1 !       NONE
;  165          1276  1 !
;  166          1277  1 ! SIDE EFFECTS:
;  167          1278  1 !
;  168          1279  1 !       Signals:
;  169          1280  1 !       BAS$K_IO_CHANOT (I/O channel not open)
;  170          1281  1 !       BAS$K_ILLILLACC (illegal or illogical access)
;  171          1282  1 !
;  172          1283  1 !--
;  173          1284  1
;  174          1285  2     BEGIN
;  175          1286  2
;  176          1287  2     BUILTIN
;  177          1288  2         FP;
;  178          1289  2
;  179          1290  2     GLOBAL REGISTER
;  180          1291  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  181          1292  2
;  182          1293  2     LOCAL
;  183          1294  2         FMP : REF BLOCK [, BYTE];
;  184          1295  2
;  185          1296  2     FMP = .FP;
;  186          1297  2 !+
;  187          1298  2 ! Allocate the LUB/ISB/RAB for this unit if necessary.  Store new CB (con-
;  188          1299  2 ! trol block) in OTS$$A_CUR_LUB.  Store signed unit number in LUB$W_LUN.
;  189          1300  2 !-
;  190          1301  2     BAS$$CB_PUSH (.UNIT, LUB$K_ILUN_MIN);
;  191          1302  2     CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
;  192          1303  2 !+
;  193          1304  2 ! If the channel is not open, give an error message.
```

```
;  194          1305  2 ! Channel 0 is not valid.
;  195          1306  2 !-
;  196          1307  2
;  197          1308  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
;  198          1309  2
;  199          1310  2 !+
;  200          1311  2 ! Now that the data base is in place, store the statement type and  go
;  201          1312  2 ! directly to the REC level.
;  202          1313  2 !-
;  203          1314  2     CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_RES;
;  204          1315  2 !+
;  205          1316  2 ! Check for virtual array usage and set block usage
;  206          1317  2 !-
;  207          1318  2     IF .CCB [LUB$V_VA_USE] THEN BAS$$STOP_IO(BAS$K_ILLILLACC);
;  208          1319  2     CCB [LUB$V_BLK_USE] = 1;
;  209          1320  2     BAS$$REC_RSE ();
;  210          1321  2 !+
;  211          1322  2 ! Now that the RESTORE has been done, pop the CCB off the I/O system.
;  212          1323  2 !-
;  213          1324  2     BAS$$CB_POP ();
;  214          1325  1     END;                                    !End of BAS$RESTORE

                                                .TITLE  BAS$RESTORE
                                                .IDENT  \1-005\

                                                .EXTRN  BAS$$REC_RIN, BAS$$REC_RSE
                                                .EXTRN  BAS$$CB_PUSH, BAS$$CB_POP
                                                .EXTRN  BAS$$STOP, BAS$$STOP_IO
                                                .EXTRN  BAS$HANDLER, BAS$K_PROLOSSOR
                                                .EXTRN  BAS$K_ILLILLACC
                                                .EXTRN  BAS$K_IO_CHANOT

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

                                 083C 00000     .ENTRY  BAS$RESTORE, Save R2,R3,R4,R5,R11       ; 1248
                54 00000000G    00   9E 00002     MOVAB   BAS$$STOP_IO, R4                        ;
                53              5D   D0 00009     MOVL    FP, FMP                                 ; 1296
                50              08   CE 0000C     MNEGL   #8, R0                                  ; 1301
                52        04    AC   D0 0000F     MOVL    UNIT, R2                                ;
                   00000000G    00   16 00013     JSB     BAS$$CB_PUSH                            ;
            FF4C  CB           0C   A3 D0 00019     MOVL    12(FMP), -180(CCB)                   ; 1302
                  07           FC   AB E8 0001F     BLBS    -4(CCB), 1$                          ; 1308
                  7E           00G  8F 9A 00023     MOVZBL  #BAS$K_IO_CHANOT, -(SP)              ;
                  64                01 FB 00027     CALLS   #1, BAS$$STOP_IO                     ;
            FF71  CB                25 90 0002A 1$:  MOVB    #37, -143(CCB)                       ; 1314
                  07           FF   AB E9 0002F     BLBC    -1(CCB), 2$                          ; 1318
                  7E           00G  8F 9A 00033     MOVZBL  #BAS$K_ILLILLACC, -(SP)              ;
                  64                01 FB 00037     CALLS   #1, BAS$$STOP_IO                     ;
              FF  AB                02 88 0003A 2$:  BISB2   #2, -1(CCB)                          ; 1319
                   00000000G    00   16 0003E     JSB     BAS$$REC_RSE                         ; 1320
                   00000000G    00   16 00044     JSB     BAS$$CB_POP                          ; 1324
                                04 0004A     RET                                                  ; 1325

; Routine Size:  75 bytes,    Routine Base:  _BAS$CODE + 0000
```

; 215 1326 1

```
;  217          1327  1 GLOBAL ROUTINE BAS$RESTORE_KEY (                      ! RESTORE indexed
;  218          1328  1         UNIT,                                        ! logical unit number
;  219          1329  1         KEY_NO                                       ! key number
;  220          1330  1     ) : NOVALUE =
;  221          1331  1
;  222          1332  1 !++
;  223          1333  1 ! FUNCTIONAL DESCRIPTION:
;  224          1334  1 !
;  225          1335  1 !       This routine will set up the I/O data base for this LUN if necessary
;  226          1336  1 !       and then go directly to the REC level.  When control is returned to
;  227          1337  1 !       this routine, it pops the CCB off of the I/O system.  The actual inter-
;  228          1338  1 !       face to RMS is done at the REC level.  The file is rewound based on the
;  229          1339  1 !       key specified.
;  230          1340  1 !
;  231          1341  1 ! FORMAL PARAMETERS:
;  232          1342  1 !
;  233          1343  1 !       UNIT.rlu.v      logical unit number
;  234          1344  1 !       KEY_NO.rlu.v    key of reference number
;  235          1345  1 !
;  236          1346  1 ! IMPLICIT INPUTS:
;  237          1347  1 !
;  238          1348  1 !       LUB$V_VA_USE            virtual array use of file
;  239          1349  1 !
;  240          1350  1 ! IMPLICIT OUTPUTS:
;  241          1351  1 !
;  242          1352  1 !       ISB$B_STTM_TYPE         the statement type
;  243          1353  1 !       LUB$V_BLK_USE           non-virtual use of file
;  244          1354  1 !
;  245          1355  1 ! COMPLETION CODES:
;  246          1356  1 !
;  247          1357  1 !       NONE
;  248          1358  1 !
;  249          1359  1 ! SIDE EFFECTS:
;  250          1360  1 !
;  251          1361  1 !       Signals:
;  252          1362  1 !       BAS$K_IO_CHANOT (I/O channel not open)
;  253          1363  1 !       BAS$K_ILLILLACC (illegal or illogical access)
;  254          1364  1 !
;  255          1365  1 !--
;  256          1366  1
;  257          1367  2     BEGIN
;  258          1368  2
;  259          1369  2     BUILTIN
;  260          1370  2         FP;
;  261          1371  2
;  262          1372  2     GLOBAL REGISTER
;  263          1373  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  264          1374  2
;  265          1375  2     LOCAL
;  266          1376  2         FMP : REF BLOCK [, BYTE];
;  267          1377  2
;  268          1378  2     FMP = .FP;
;  269          1379  2 !+
;  270          1380  2 ! Allocate the LUB/ISB/RAB for this unit if necessary.  Store new CB (con-
;  271          1381  2 ! trol block) in OTS$$A_CUR_LUB.  Store signed unit number in LUB$W_LUN.
;  272          1382  2 !-
;  273          1383  2     BAS$$CB_PUSH (.UNIT, LUB$K_ILUN_MIN);
```

```
;  274          1384  2     CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
;  275          1385  2 !+
;  276          1386  2 ! If the channel is not open, give an error message.
;  277          1387  2 !-
;  278          1388  2
;  279          1389  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
;  280          1390  2
;  281          1391  2 !+
;  282          1392  2 ! Now that the data base is in place, store the statement type and  go
;  283          1393  2 ! directly to the REC level.
;  284          1394  2 !-
;  285          1395  2     CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_RIN;
;  286          1396  2 !+
;  287          1397  2 ! Check for virtual array usage and set block usage
;  288          1398  2 !-
;  289          1399  2     IF .CCB [LUB$V_VA_USE] THEN BAS$$STOP_IO(BAS$K_ILLILLACC);
;  290          1400  2     CCB [LUB$V_BLK_USE] = 1;
;  291          1401  2     BAS$$REC_RIN (.KEY_NO);
;  292          1402  2 !+
;  293          1403  2 ! Now that the GET has been done, pop the CCB off the I/O system.
;  294          1404  2 !-
;  295          1405  2     BAS$$CB_POP ();
;  296          1406  1     END;


                                        083C 00000          .ENTRY  BAS$RESTORE_KEY, Save R2,R3,R4,R5,R11       ; 1327
                    54 00000000G    00  9E 00002          MOVAB   BAS$$STOP_IO, R4                            ;
                    53              5D  D0 00009          MOVL    FP, FMP                                     ; 1378
                    50              08  CE 0000C          MNEGL   #8, R0                                      ; 1383
                    52        04    AC  D0 0000F          MOVL    UNIT, R2                                    ;
                       00000000G    00  16 00013          JSB     BAS$$CB_PUSH                                ;
                FF4C   CB        0C  A3  D0 00019          MOVL    12(FMP), -180(CCB)                          ; 1384
                    07           FC  AB  E8 0001F          BLBS    -4(CCB), 1$                                 ; 1389
                    7E           00G 8F  9A 00023          MOVZBL  #BAS$K_IO_CHANOT, -(SP)                     ;
                    64              01  FB 00027          CALLS   #1, BAS$$STOP_IO                            ;
                FF71   CB           31  90 0002A 1$:      MOVB    #49, -143(CCB)                              ; 1395
                    07           FF  AB  E9 0002F          BLBC    -1(CCB), 2$                                 ; 1399
                    7E           00G 8F  9A 00033          MOVZBL  #BAS$K_ILLILLACC, -(SP)                     ;
                    64              01  FB 00037          CALLS   #1, BAS$$STOP_IO                            ;
                      FF        AB  02  88 0003A 2$:      BISB2   #2, -1(CCB)                                 ; 1400
                    50        08    AC  D0 0003E          MOVL    KEY_NO, R0                                  ; 1401
                       00000000G    00  16 00042          JSB     BAS$$REC_RIN                                ;
                       00000000G    00  16 00048          JSB     BAS$$CB_POP                                 ; 1405
                                        04 0004E          RET                                                 ; 1406

; Routine Size:  79 bytes,    Routine Base:  _BAS$CODE + 004B

;  297          1407  1
```

```
;  299     1408  1 GLOBAL ROUTINE BAS$RESTORE_DAT : NOVALUE =      ! Restore DATA pointer
;  300     1409  1
;  301     1410  1 !++
;  302     1411  1 ! FUNCTIONAL DESCRIPTION:
;  303     1412  1 !
;  304     1413  1 !       Restore the current DATA pointer, so that it again points
;  305     1414  1 !       to the beginning of the DATA text.  This routine is called
;  306     1415  1 !       by the RESTORE BASIC statement.
;  307     1416  1 !
;  308     1417  1 ! FORMAL PARAMETERS:
;  309     1418  1 !
;  310     1419  1 !       NONE
;  311     1420  1 !
;  312     1421  1 ! IMPLICIT INPUTS:
;  313     1422  1 !
;  314     1423  1 !       BAS$L_IN_BEG_DA.rl      Offset to the beginning of the DATA
;  315     1424  1 !                               string (in the argument list to the
;  316     1425  1 !                               routine that initialized the frame)
;  317     1426  1 !       BSF$L_INIT_REL.rl       Base for the above offset
;  318     1427  1 !       BSF$A_INIT_ARG.ra       Address of the arg list (which contains
;  319     1428  1 !                               BAS$L_IN_BEG_DA)
;  320     1429  1 !
;  321     1430  1 ! IMPLICIT OUTPUTS:
;  322     1431  1 !
;  323     1432  1 !       BSF$A_CUR_DTA.wa        Current position in the DATA string
;  324     1433  1 !
;  325     1434  1 ! ROUTINE VALUE:
;  326     1435  1 ! COMPLETION CODES:
;  327     1436  1 !
;  328     1437  1 !       NONE
;  329     1438  1 !
;  330     1439  1 ! SIDE EFFECTS:
;  331     1440  1 !
;  332     1441  1 !       Manipulates the frame of its caller, which must be a BASIC frame.
;  333     1442  1 !
;  334     1443  1 !--
;  335     1444  1
;  336     1445  2     BEGIN
;  337     1446  2
;  338     1447  2     BUILTIN
;  339     1448  2         FP;
;  340     1449  2
;  341     1450  2     LOCAL
;  342     1451  2         FMP : REF BLOCK [0, BYTE] FIELD (BSF$FCD),      ! our frame
;  343     1452  2         CALLERS_FCD : REF BLOCK [0, BYTE] FIELD (BSF$FCD),      ! caller's frame
;  344     1453  2         INIT_ARG : REF BLOCK [0, BYTE] FIELD (BAS$INIT_ARGS),   ! init arg list
;  345     1454  2         MAJOR_FCD : REF BLOCK [0, BYTE] FIELD (BSF$FCD);        ! caller's major frame
;  346     1455  2
;  347     1456  2 !+
;  348     1457  2 ! Get pointer to caller's frame.
;  349     1458  2 !-
;  350     1459  2     FMP = .FP;
;  351     1460  2     CALLERS_FCD = .FMP [BSF$A_SAVED_FP];
;  352     1461  2 !+
;  353     1462  2 ! If the caller's frame is not a BASIC frame, we have a fatal error.
;  354     1463  2 !-
;  355     1464  2
```

```
;  356          1465  2     IF (.CALLERS_FCD [BSF$A_HANDLER] NEQA BAS$HANDLER) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;  357          1466  2
;  358          1467  2 !+
;  359          1468  2 ! Get a pointer to the caller's major frame.  This may be the same as
;  360          1469  2 ! the caller's frame, but it will be different if, for example, the
;  361          1470  2 ! RESTORE is in a GOSUB, DEF or condition handler.
;  362          1471  2 !-
;  363          1472  2     MAJOR_FCD = .CALLERS_FCD [BSF$A_BASE_R11] + %FIELDEXPAND (BSF$FRAME_BASE, 0);
;  364          1473  2 !+
;  365          1474  2 ! Get a pointer to the initialization argument list for the caller's
;  366          1475  2 ! frame.  The initialization routine cleverly saved such a pointer
;  367          1476  2 ! for us in the major frame.
;  368          1477  2 !-
;  369          1478  2     INIT_ARG = .MAJOR_FCD [BSF$A_INIT_ARG];
;  370          1479  2 !+
;  371          1480  2 ! Relocate the offset to the beginning of the DATA text, and store
;  372          1481  2 ! it in the frame as the current pointer to the DATA text.
;  373          1482  2 !-
;  374          1483  2     MAJOR_FCD [BSF$A_CUR_DATA] = .INIT_ARG [BAS$L_IN_BEG_DA] + .MAJOR_FCD [BSF$L_INIT_REL];
;  375          1484  2 !+
;  376          1485  2 ! All done.
;  377          1486  2 !-
;  378          1487  1     END;                                    ! end of BAS$RESTORE_DAT


                                                0004 00000          .ENTRY  BAS$RESTORE_DAT, Save R2                ; 1408
                                    50          5D  D0 00002        MOVL    FP, FMP                                 ; 1459
                                    52      0C  A0  D0 00005        MOVL    12(FMP), CALLERS_FCD                    ; 1460
                                    50 00000000G    00  9E 00009    MOVAB   BAS$HANDLER, R0                         ; 1465
                                    50              62  D1 00010    CMPL    (CALLERS_FCD), R0                       ;
                                                    0B  13 00013    BEQL    1$                                      ;
                                    7F      00G     8F  9A 00015    MOVZBL  #BAS$K_PROLOSSOR, -(SP)                 ;
                            00000000G       00      01  FB 00019    CALLS   #1, BAS$$STOP                           ;
                        50      F4      A2 000000C3     8F  C1 00020 1$: ADDL3   #195, -12(CALLERS_FCD), MAJOR_FCD   ; 1472
                                    51      D8  A0  D0 00029        MOVL    -40(MAJOR_FCD), INIT_ARG                ; 1478
                    C4  A0      38  A1      DC  A0  C1 0002D        ADDL3   -36(MAJOR_FCD), 56(INIT_ARG), -         ; 1483
                                                                            -60(MAJOR_FCD)                          ;
                                                04 00034            RET                                             ; 1487

; Routine Size:  53 bytes,    Routine Base:  _BAS$CODE + 009A


;  379          1488  1
;  380          1489  1 END                                     !End of module - BAS$RESTORE
;  381          1490  1
;  382          1491  0 ELUDOM



;                                   PSECT SUMMARY
;
```

```
;       Name                    Bytes                           Attributes
;
;  _BAS$CODE                     207  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         1         0       581          00:01.2
```

```
;                                   COMMAND QUALIFIERS

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

; Size:         207 code + 0 data bytes
; Run Time:       00:12.2
; Elapsed Time:   00:28.5
; Lines/CPU Min:   7350
; Lexemes/CPU-Min: 38913
; Memory Used:  130 pages
; Compilation Complete
```

BASRSTSUB
LIS

BASRAD50
LIS

BASRSET
LIS

BASPUT
LIS

BASRECPRO
LIS

BASRESTAR
LIS

BASRANDOM
LIS

BASREMAP
LIS

BASRESTOR
LIS

BASRIGHT
LIS