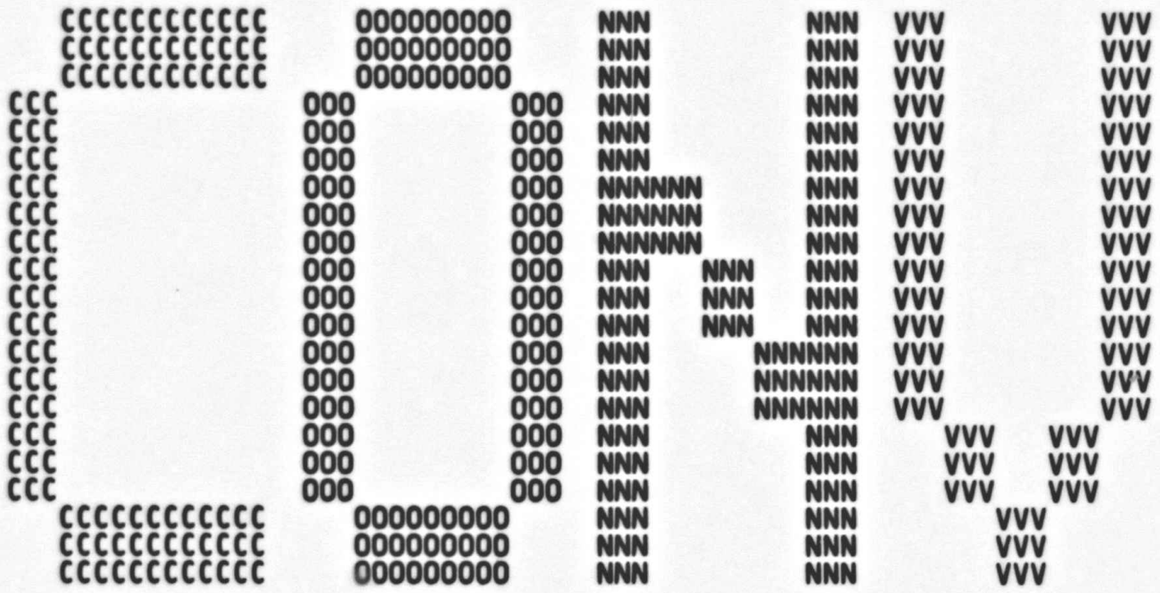
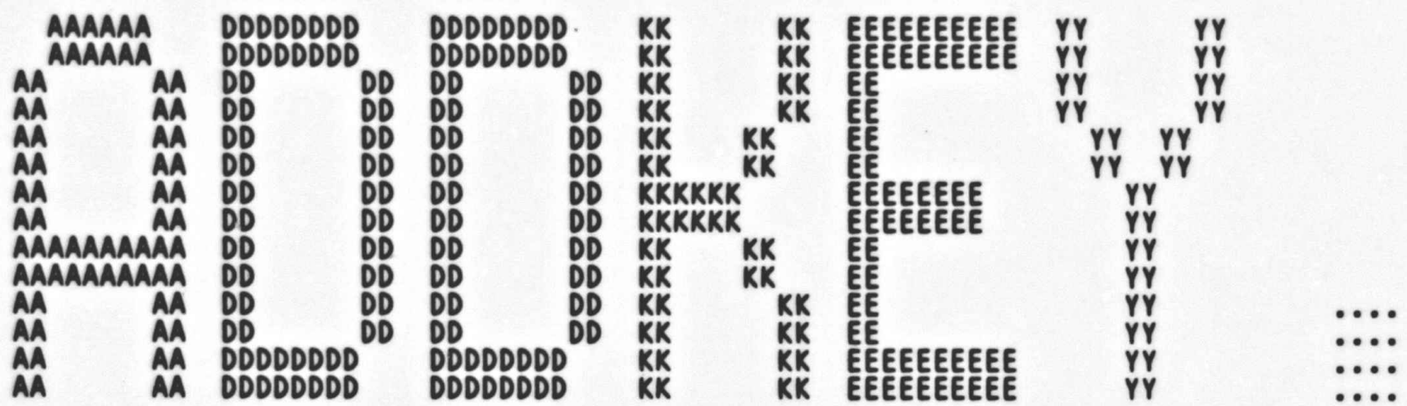
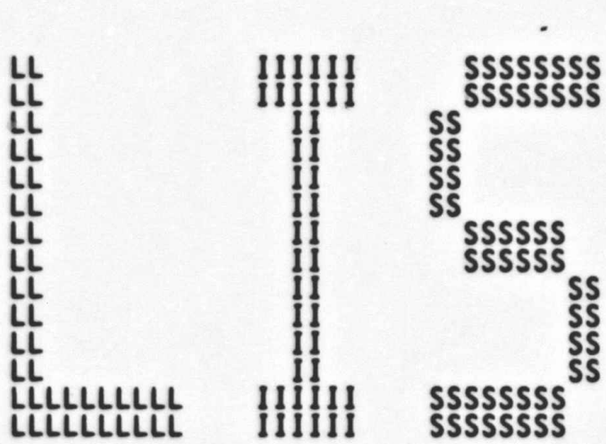

CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCCC OOOOOOOOO NNN NNN VVV

FILEID**ADDKEY





```
;  1          0001  0 %TITLE  'VAX-11 CONVERT'
;  2          0002  0 MODULE  ADD$KEY          ( IDENT='V04-000',
;  3          0003  0                          OPTLEVEL=3
;  4          0004  0                          ) =
;  5          0005  0
;  6          0006  1 BEGIN
;  7          0007  1
;  8          0008  1 !*** ****************************************************************************
;  9          0009  1 !*                                                                              *
; 10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                    *
; 11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
; 12          0012  1 !*  ALL RIGHTS RESERVED.                                                       *
; 13          0013  1 !*                                                                            *
; 14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
; 15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
; 16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
; 17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
; 18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
; 19          0019  1 !*  TRANSFERRED.                                                               *
; 20          0020  1 !*                                                                            *
; 21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
; 22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
; 23          0023  1 !*  CORPORATION.                                                               *
; 24          0024  1 !*                                                                            *
; 25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
; 26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
; 27          0027  1 !*                                                                            *
; 28          0028  1 !*                                                                            *
; 29          0029  1 !****************************************************************************
```

```
;     31        0030  1 !++
;     32        0031  1 !
;     33        0032  1 ! Facility:     VAX-11 CONVERT
;     34        0033  1 !
;     35        0034  1 ! Abstract:     ADD_KEY routines
;     36        0035  1 !
;     37        0036  1 ! Contents:
;     38        0037  1 !                   CHECK_KEY
;     39        0038  1 !                   MAKE_DESCRIPTOR
;     40        0039  1 !                   CHECK_XAB
;     41        0040  1 !                   STUFF_KEY_DESC
;     42        0041  1 !                   LOAD_KEY
;     43        0042  1 !
;     44        0043  1 ! Environment:
;     45        0044  1 !
;     46        0045  1 !                   VAX/VMS Operating System
;     47        0046  1 !
;     48        0047  1 !--
;     49        0048  1
;     50        0049  1 !
;     51        0050  1 ! Author:       Keith B. Thompson       Creation date:  December-1982
;     52        0051  1 !
;     53        0052  1 !
;     54        0053  1 ! Modified by:
;     55        0054  1 !
;     56        0055  1 !****
```

```
;    58          0056  1
;    59          0057  1 PSECT
;    60          0058  1         OWN     = _CONV$OWN     (PIC),
;    61          0059  1         GLOBAL  = _CONV$GLOBAL  (PIC),
;    62          0060  1         PLIT    = _CONV$PLIT    (SHARE,PIC),
;    63          0061  1         CODE    = _CONV$CODE    (SHARE,PIC);
;    64          0062  1
;    65          0063  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    66          0064  1 LIBRARY 'SRC$:CONVERT';
;    67          0065  1
;    68          0066  1 !       Error codes
;    69          0067  1 !
;    70          0068  1 DEFINE_ERROR_CODES;
;    71          0069  1
;    72          0070  1 EXTERNAL ROUTINE
;    73          0071  1         CONV$$EXTEND_AREA       : CL$EXTEND_AREA        NOVALUE,
;    74          0072  1         CONV$$GET_VM            : CL$GET_VM,
;    75          0073  1         CONV$$INIT_FAST_LOAD    : CL$INIT_FAST_LOAD,
;    76          0074  1         CONV$$LOAD_SECONDARY    : CL$LOAD_SECONDARY     NOVALUE,
;    77          0075  1         CONV$$PARSE_DEF,
;    78          0076  1         CONV$$READ_PROLOGUE     : CL$READ_PROLOGUE      NOVALUE,
;    79          0077  1         CONV$$RMS_OPEN_ERROR    : NOVALUE,
;    80          0078  1         CONV$$SET_KEY_DESC      : CL$SET_KEY_DESC,
;    81          0079  1         CONV$$SORT_SECONDARY    : CL$SORT_SECONDARY,
;    82          0080  1         CONV$$WRITE_AREA_DESC   : CL$WRITE_AREA_DESC    NOVALUE,
;    83          0081  1         CONV$$WRITE_KEY_DESC    : CL$WRITE_KEY_DESC     NOVALUE,
;    84          0082  1         CONV$$WRITE_PROLOGUE    : NOVALUE;
;    85          0083  1
;    86          0084  1 MACRO
;    87        M 0085  1         DEFINE_XAB_GLOBAL       = GLOBAL REGISTER
;    88          0086  1                                       XAB = 10 : REF BLOCK [ ,BYTE ]%,
;    89          0087  1
;    90        M 0088  1         DEFINE_XAB              = EXTERNAL REGISTER
;    91          0089  1                                       XAB : REF BLOCK [ ,BYTE ]%;
;    92          0090  1
;    93          0091  1 LINKAGE
;    94          0092  1         AL$MAKE_DESCRIPTOR      = JSB : GLOBAL ( XAB            = 10,
;    95          0093  1                                                  KEY_DESC       = 11 ),
;    96          0094  1         AL$CHECK_XAB            = JSB : GLOBAL ( XAB            = 10,
;    97          0095  1                                                  KEY_DESC       = 11 ),
;    98          0096  1         AL$STUFF_KEY_DESC       = JSB : GLOBAL ( XAB            = 10,
;    99          0097  1                                                  KEY_DESC       = 11 );
;   100          0098  1
;   101          0099  1 FORWARD ROUTINE
;   102          0100  1         MAKE_DESCRIPTOR         : AL$MAKE_DESCRIPTOR    NOVALUE,
;   103          0101  1         CHECK_XAB               : AL$CHECK_XAB,
;   104          0102  1         STUFF_KEY_DESC          : AL$STUFF_KEY_DESC     NOVALUE;
;   105          0103  1
;   106          0104  1 EXTERNAL
;   107          0105  1         CONV$AB_FLAGS           : BLOCK [ ,BYTE ],
;   108          0106  1         CONV$AR_OUT_FILE_NAM    : REF DESC_BLK,                 ! Output File
;   109          0107  1
;   110          0108  1         CONV$AB_OUT_XABSUM      : $XABSUM_DECL,
;   111          0109  1         CONV$AB_OUT_NAM         : $NAM_DECL,
;   112          0110  1         CONV$AB_OUT_FAB         : $FAB_DECL,
;   113          0111  1         CONV$AB_OUT_RAB         : $RAB_DECL,
;   114          0112  1
```

```
;  115     0113  1          CONV$AB_FDL_FAB         : REF BLOCK [ ,BYTE ],
;  116     0114  1          CONV$AB_FDL_RAB         : REF BLOCK [ ,BYTE ],
;  117     0115  1
;  118     0116  1          CONV$GL_WORK_F,
;  119     0117  1
;  120     0118  1          CONV$GL_KEY_DESC_BUF,
;  121     0119  1          CONV$GL_KEY_DESC_VBN,
;  122     0120  1          CONV$AR_AREA_BLOCK      : REF BLOCKVECTOR [ ,AREA$C_BLN,BYTE ];
;  123     0121  1
;  124     0122  1 GLOBAL
;  125     0123  1          CONV$GL_ADD_DELE_KEY    : LONG;
;  126     0124  1
```

```
;  128     0125  1 %SBTTL  'OPEN_OUTPUT'
;  129     0126  1 GLOBAL ROUTINE  ADD$$OPEN_OUTPUT =
;  130     0127  1 !++
;  131     0128  1 !
;  132     0129  1 ! Functional Description:
;  133     0130  1 !
;  134     0131  1 !       Opens the output file, connects a record stream and
;  135     0132  1 !       allocates and fills in the prologue key and
;  136     0133  1 !       area descriptor blocks for sort and or fast load
;  137     0134  1 !
;  138     0135  1 ! Calling Sequence:
;  139     0136  1 !
;  140     0137  1 !       ADD$$OPEN_OUTPUT
;  141     0138  1 !
;  142     0139  1 ! Input Parameters:
;  143     0140  1 !       none
;  144     0141  1 !
;  145     0142  1 ! Implicit Inputs:
;  146     0143  1 !
;  147     0144  1 !       CONV$AB_OUT_FAB - Output fab
;  148     0145  1 !
;  149     0146  1 ! Output Parameters:
;  150     0147  1 !       none
;  151     0148  1 !
;  152     0149  1 ! Implicit Outputs:
;  153     0150  1 !
;  154     0151  1 !       CONV$AB_FLAGS [ CONV$V_OUT ]    - Set on success
;  155     0152  1 !
;  156     0153  1 ! Routine Value:
;  157     0154  1 !
;  158     0155  1 !       CONV$_SUCCESS or error
;  159     0156  1 !
;  160     0157  1 ! Routines Called:
;  161     0158  1 !
;  162     0159  1 !       $PARSE
;  163     0160  1 !       CONV$$RMS_OPEN_ERROR    - By RMS as an AST
;  164     0161  1 !       $DISPLAY
;  165     0162  1 !       $OPEN
;  166     0163  1 !       CONV$$READ_PROLOGUE
;  167     0164  1 !       $CONNECT
;  168     0165  1 !
;  169     0166  1 ! Side Effects:
;  170     0167  1 !       none
;  171     0168  1 !
;  172     0169  1 !--
;  173     0170  1
;  174     0171  2     BEGIN
;  175     0172  2
;  176     0173  2     ! Any rms errors on the output fab are OPENOUT errors
;  177     0174  2     !
;  178     0175  2 !    CONV$AB_OUT_FAB [ FAB$L_CTX ] = CONV$_OPENOUT;
;  179     0176  2
;  180     0177  2     ! Use the file name in the call argument
;  181     0178  2     !
;  182     0179  2     CONV$AB_OUT_FAB [ FAB$B_FNS ] = .CONV$AR_OUT_FILE_NAM [ DSC$W_LENGTH ];
;  183     0180  2     CONV$AB_OUT_FAB [ FAB$L_FNA ] = .CONV$AR_OUT_FILE_NAM [ DSC$A_POINTER ];
;  184     0181  2
```

```
;  185     0182  2     $OPEN( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
;  186     0183  2
;  187     0184  2     ! If we got here then we have opened a file.
;  188     0185  2     !
;  189     0186  2     CONV$AB_FLAGS [ CONV$V_OUT ] = _SET;
;  190     0187  2
;  191     0188  2     ! Is't not very exciting if it's not an index file
;  192     0189  2     !
;  193     0190  2     IF .CONV$AB_OUT_FAB [ FAB$B_ORG ] NEQU FAB$C_IDX
;  194     0191  2     THEN
;  195     0192  2         RETURN CONV$_NOTIDX;
;  196     0193  2
;  197     0194  2     ! Connect the file for Block IO for reading the prologue.
;  198     0195  2     !
;  199     0196  2     $CONNECT ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_OPEN_ERROR );
;  200     0197  2
;  201     0198  2     ! Read the prologue
;  202     0199  2     !
;  203     0200  2     CONV$$READ_PROLOGUE();
;  204     0201  2
;  205     0202  2     ! Any errors on the output rab should be write errors (exceptions are in
;  206     0203  2     ! the fast load code
;  207     0204  2     !
;  208     0205  2     CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
;  209     0206  2
;  210     0207  2     ! Return normally
;  211     0208  2     !
;  212     0209  2     RETURN CONV$_SUCCESS
;  213     0210  2
;  214     0211  1     END;


                                .TITLE  ADD$KEY VAX-11 CONVERT
                                .IDENT  \V04-000\

                                .PSECT  _CONV$GLOBAL,NOEXE,  PIC,2

00000 CONV$GL_ADD_DELE_KEY::
                                .BLKB   4

                                .EXTRN  CONVERT$_FACILITY
                                .EXTRN  CONV$_FAO_MAX, CONV$_BADBLK
                                .EXTRN  CONV$_BADLOGIC, CONV$_BADSORT
                                .EXTRN  CONV$_CONFQUAL, CONV$_CREATEDSTM
                                .EXTRN  CONV$_CREA_ERR, CONV$_DELPRI
                                .EXTRN  CONV$_DUP, CONV$_EXTN_ERR
                                .EXTRN  CONV$_FATALEXC, CONV$_FILLIM
                                .EXTRN  CONV$_IDX_LIM, CONV$_ILL_KEY
                                .EXTRN  CONV$_ILL_VALUE
                                .EXTRN  CONV$_INP_FILES
                                .EXTRN  CONV$_INSVIRMEM
                                .EXTRN  CONV$_INVBKT, CONV$_KEY
                                .EXTRN  CONV$_KEYREF, CONV$_LOADIDX
                                .EXTRN  CONV$_NARG, CONV$_NI
                                .EXTRN  CONV$_NOKEY, CONV$_NOTIDX
                                .EXTRN  CONV$_NOTSEQ, CONV$_NOWILD
                                .EXTRN  CONV$_ORDER, CONV$_OPENEXC
```

```
                                                .EXTRN  CONV$_OPENIN, CONV$_OPENOUT
                                                .EXTRN  CONV$_PAD, CONV$_PLV
                                                .EXTRN  CONV$_PROERR, CONV$_PROL_WRT
                                                .EXTRN  CONV$_READERR, CONV$_RSK
                                                .EXTRN  CONV$_RSZ, CONV$_RTL
                                                .EXTRN  CONV$_RTS, CONV$_SEQ
                                                .EXTRN  CONV$_UDF_BKS, CONV$_UDF_BLK
                                                .EXTRN  CONV$_VFC, CONV$_WRITEERR
                                                .EXTRN  CONV$$EXTEND_AREA
                                                .EXTRN  CONV$$GET_VM, CONV$$INIT_FAST_LOAD
                                                .EXTRN  CONV$$LOAD_SECONDARY
                                                .EXTRN  CONV$$PARSE_DEF
                                                .EXTRN  CONV$$READ_PROLOGUE
                                                .EXTRN  CONV$$RMS_OPEN_ERROR
                                                .EXTRN  CONV$$SET_KEY_DESC
                                                .EXTRN  CONV$$SORT_SECONDARY
                                                .EXTRN  CONV$$WRITE_AREA_DESC
                                                .EXTRN  CONV$$WRITE_KEY_DESC
                                                .EXTRN  CONV$$WRITE_PROLOGUE
                                                .EXTRN  CONV$AB_FLAGS, CONV$AR_OUT_FILE_NAM
                                                .EXTRN  CONV$AB_OUT_XABSUM
                                                .EXTRN  CONV$AB_OUT_NAM
                                                .EXTRN  CONV$AB_OUT_FAB
                                                .EXTRN  CONV$AB_OUT_RAB
                                                .EXTRN  CONV$AB_FDL_FAB
                                                .EXTRN  CONV$AB_FDL_RAB
                                                .EXTRN  CONV$GL_WORK_F, CONV$GL_KEY_DESC_BUF
                                                .EXTRN  CONV$GL_KEY_DESC_VBN
                                                .EXTRN  CONV$AR_AREA_BLOCK
                                                .EXTRN  SYS$OPEN, SYS$CONNECT

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

                                    0FFC 00000  .ENTRY  ADD$$OPEN_OUTPUT, Save R2,R3,R4,R5,R6,R7,-  ; 0126
                                                        R8,R9,R10,R11
                50      0000G   CF  D0 00002        MOVL   CONV$AR_OUT_FILE_NAM, R0                 ; 0179
          0000G CF              60  90 00007        MOVB   (R0), CONV$AB_OUT_FAB+52
          0000G CF        04    A0  D0 0000C        MOVL   4(R0), CONV$AB_OUT_FAB+44                ; 0180
                        0000G   CF  9F 00012        PUSHAB CONV$$RMS_OPEN_ERROR                     ; 0182
                        0000G   CF  9F 00016        PUSHAB CONV$AB_OUT_FAB
00000000G       00              02  FB 0001A        CALLS  #2, SYS$OPEN
          0000G CF              02  88 00021        BISB2  #2, CONV$AB_FLAGS+2                      ; 0186
                20      0000G   CF  91 00026        CMPB   CONV$AB_OUT_FAB+29, #32                  ; 0190
                                08  13 0002B        BEQL   1$
                50 00000000G    8F  D0 0002D        MOVL   #CONV$_NOTIDX, R0                        ; 0192
                                    04 00034        RET
                        0000G   CF  9F 00035 1$:    PUSHAB CONV$$RMS_OPEN_ERROR                     ; 0196
                        0000G   CF  9F 00039        PUSHAB CONV$AB_OUT_RAB
00000000G       00              02  FB 0003D        CALLS  #2, SYS$CONNECT
                              0000G 30  00044       BSBW   CONV$$READ_PROLOGUE                      ; 0200
          0000G CF 00000000G    8F  D0 00047        MOVL   #CONV$_WRITEERR, CONV$AB_OUT_RAB+24      ; 0205
                50              01  D0 00050        MOVL   #1, R0                                   ; 0209
                                    04 00053        RET                                             ; 0211

; Routine Size:  84 bytes,    Routine Base:  _CONV$CODE + 0000
```

```
;  216     0212  1 %SBTTL  'CHECK_KEY'
;  217     0213  1 GLOBAL ROUTINE  ADD$$CHECK_KEY : AL$CHECK_KEY =
;  218     0214  1 !++
;  219     0215  1 !
;  220     0216  1 ! Functional Description:
;  221     0217  1 !
;  222     0218  1 !       This routine parses the fdl file and searches for the proper key xab
;  223     0219  1 !       if one is found it is verified and a index descriptor is added to the
;  224     0220  1 !       file
;  225     0221  1 !
;  226     0222  1 ! Calling Sequence:
;  227     0223  1 !
;  228     0224  1 !       ADD$$CHECK_KEY()
;  229     0225  1 !
;  230     0226  1 ! Input Parameters:
;  231     0227  1 !       none
;  232     0228  1 !
;  233     0229  1 ! Implicit Inputs:
;  234     0230  1 !
;  235     0231  1 !
;  236     0232  1 ! Output Parameters:
;  237     0233  1 !       none
;  238     0234  1 !
;  239     0235  1 ! Implicit Outputs:
;  240     0236  1 !
;  241     0237  1 !
;  242     0238  1 ! Routine Value:
;  243     0239  1 !
;  244     0240  1 !
;  245     0241  1 ! Routines Called:
;  246     0242  1 !
;  247     0243  1 !
;  248     0244  1 ! Side Effects:
;  249     0245  1 !       none
;  250     0246  1 !
;  251     0247  1 !--
;  252     0248  1
;  253     0249  2     BEGIN
;  254     0250  2
;  255     0251  2     DEFINE_XAB_GLOBAL;
;  256     0252  2     DEFINE_KEY_DESC;
;  257     0253  2
;  258     0254  2     LOCAL
;  259     0255  2         STATUS;
;  260     0256  2
;  261     0257  2     ! Check the key against the file
;  262     0258  2     !
;  263     0259  2     IF .CONV$GL_ADD_DELE_KEY NEQU .CONV$AB_OUT_XABSUM [ XAB$B_NOK ]
;  264     0260  2     THEN
;  265     0261  2         RETURN 0;       ! illegal key value
;  266     0262  2
;  267     0263  2     ! Find the key xab described by the fdl file (first parse it)
;  268     0264  2     !
;  269     0265  3     IF NOT ( STATUS = CONV$$PARSE_DEF() )
;  270     0266  2     THEN
;  271     0267  2         RETURN .STATUS;
;  272     0268  2
```

```
;   273         0269  2     ! Get the first xab
;   274         0270  2     !
;   275         0271  2     XAB = .CONV$AB_FDL_FAB [ FAB$L_XAB ];
;   276         0272  2
;   277         0273  2     ! Loop until xabs are exausted or the correct one is found
;   278         0274  2     !
;   279         0275  2     WHILE .XAB NEQU 0
;   280         0276  2     DO
;   281         0277  2         BEGIN
;   282         0278  2
;   283         0279  2         ! Is this a key xab?
;   284         0280  2         !
;   285         0281  2         IF .XAB [ XAB$B_COD ] EQLU XAB$C_KEY
;   286         0282  2         THEN
;   287         0283  2
;   288         0284  2             ! Is it the right one?
;   289         0285  2             !
;   290         0286  2             IF .XAB [ XAB$B_REF ] EQLU .CONV$GL_ADD_DELE_KEY
;   291         0287  2             THEN
;   292         0288  2                 EXITLOOP;
;   293         0289  2
;   294         0290  2             XAB = .XAB [ XAB$L_NXT ]
;   295         0291  2
;   296         0292  2             END;
;   297         0293  2
;   298         0294  2     ! Did we find it?
;   299         0295  2     !
;   300         0296  2     IF .XAB EQLU 0
;   301         0297  2     THEN
;   302         0298  2         RETURN 0;       ! key not defined
;   303         0299  2
;   304         0300  2     ! Check the xab to see if it is a valid key
;   305         0301  2     !
;   306         0302  3     IF NOT ( STATUS = CHECK_XAB() )
;   307         0303  2     THEN
;   308         0304  2         RETURN .STATUS;
;   309         0305  2 !       SIGNAL_STOP( CONV$_PLV,.STATUS,.CONV$GL_ADD_DELE_KEY );
;   310         0306  2
;   311         0307  2     MAKE_DESCRIPTOR();
;   312         0308  2
;   313         0309  2     RETURN CONV$_SUCCESS
;   314         0310  2
;   315         0311  1     END;


                                    5A  DD 00000 ADD$$CHECK_KEY::
                                                 PUSHL   R10                                        ; 0213
0000'  CF     0000G  CF       08     00  ED 00002          CMPZV   #0, #8, CONV$AB_OUT_XABSUM+9, -    ; 0259
                                                         CONV$GL_ADD_DELE_KEY
                                    3A  12 0000B         BNEQ    4$                                 ;
                             0000G  CF                   00  FB 0000D         CALLS   #0, CONV$$PARSE_DEF        ; 0265
                                    34                   50  E9 00012         BLBC    STATUS, 5$                 ;
                                    51      0000G        CF  D0 00015         MOVL    CONV$AB_FDL_FAB, R1        ; 0271
                                    5A        24         A1  D0 0001A         MOVL    36(R1), XAB                ;
```

```
                                 15  13 0001E 1$:     BEQL    3$                                                      ; 0275
                              15    6A 91 00020      CMPB    (XAB), #21                                              ; 0281
                                    0A 12 00023      BNEQ    2$                                                      ;
    0000'  CF     17   AA       08   00 ED 00025      CMPZV   #0, #8, 23(XAB), CONV$GL_ADD_DELE_KEY                    ; 0286
                                    06 13 0002D      BEQL    3$                                                      ;
                              5A    04  AA D0 0002F 2$:     MOVL    4(XAB), XAB                                     ; 0290
                                    E9 11 00033      BRB     1$                                                      ;
                                    5A D5 00035 3$:     TSTL    XAB                                                  ; 0296
                                    0E 13 00037      BEQL    4$                                                      ;
                                 0000V 30 00039      BSBW    CHECK_XAB                                               ; 0302
                              0A    50 E9 0003C      BLBC    STATUS, 5$                                              ;
                                 0000V 30 0003F      BSBW    MAKE_DESCRIPTOR                                         ; 0307
                              50    01 D0 00042      MOVL    #1, R0                                                  ; 0309
                                    02 11 00045      BRB     5$                                                      ;
                                    50 D4 00047 4$:     CLRL    R0                                                   ; 0311
                              5A    8E D0 00049 5$:     MOVL    (SP)+, R10                                           ;
                                       05 0004C      RSB                                                             ;

; Routine Size:  77 bytes,    Routine Base:  _CONV$CODE + 0054
```

```
;  317     0312  1 %SBTTL  'MAKE_DESCRIPTOR'
;  318     0313  1 ROUTINE MAKE_DESCRIPTOR : AL$MAKE_DESCRIPTOR NOVALUE =
;  319     0314  1 !++
;  320     0315  1 !
;  321     0316  1 ! Functional Description:
;  322     0317  1 !
;  323     0318  1 !       Allocates and writes a key descriptor to the file then links
;  324     0319  1 !       it into the key descriptor chain
;  325     0320  1 !
;  326     0321  1 ! Calling Sequence:
;  327     0322  1 !
;  328     0323  1 !       MAKE_DESCRIPTOR()
;  329     0324  1 !
;  330     0325  1 ! Input Parameters:
;  331     0326  1 !       none
;  332     0327  1 !
;  333     0328  1 ! Implicit Inputs:
;  334     0329  1 !
;  335     0330  1 !
;  336     0331  1 ! Output Parameters:
;  337     0332  1 !       none
;  338     0333  1 !
;  339     0334  1 ! Implicit Outputs:
;  340     0335  1 !
;  341     0336  1 !
;  342     0337  1 ! Routine Value:
;  343     0338  1 !       none
;  344     0339  1 !
;  345     0340  1 ! Routines Called:
;  346     0341  1 !
;  347     0342  1 !
;  348     0343  1 ! Side Effects:
;  349     0344  1 !       none
;  350     0345  1 !
;  351     0346  1 !--
;  352     0347  1
;  353     0348  2     BEGIN
;  354     0349  2
;  355     0350  2     DEFINE_XAB;
;  356     0351  2     DEFINE_KEY_DESC;
;  357     0352  2
;  358     0353  2     LOCAL
;  359     0354  2         VBN;
;  360     0355  2
;  361     0356  2     ! Check to see if the new key descriptor fits if not extend the file
;  362     0357  2     !
;  363     0358  2     IF 1 GTRU ( .CONV$AR_AREA_BLOCK [ 0,AREA$L_CNBLK ] -
;  364     0359  2                 .CONV$AR_AREA_BLOCK [ 0,AREA$L_USED ] )
;  365     0360  2     THEN
;  366     0361  2         CONV$$EXTEND_AREA( 0 );
;  367     0362  2
;  368     0363  2     ! Update the area descriptor and save the vbn
;  369     0364  2     !
;  370     0365  2     CONV$GL_KEY_DESC_VBN = .CONV$AR_AREA_BLOCK [ 0,AREA$L_NXTVBN ];
;  371     0366  2
;  372     0367  2     CONV$AR_AREA_BLOCK [ 0,AREA$L_NXTVBN ] =
;  373     0368  2                         .CONV$AR_AREA_BLOCK [ 0,AREA$L_NXTVBN ] + 1;
```

```
;  374          0369  2     CONV$AR_AREA_BLOCK [ 0,AREA$L_USED ] =
;  375          0370  2                         .CONV$AR_AREA_BLOCK [ 0,AREA$L_USED ] + 1;
;  376          0371  2
;  377          0372  2     ! Update the area desc so as to remove the hole we just made
;  378          0373  2     !
;  379          0374  2     CONV$$WRITE_AREA_DESC( 0 );
;  380          0375  2
;  381          0376  2     ! Make your own key descriptor in the normal buffer (VBN already set)
;  382          0377  2     !
;  383          0378  2     KEY_DESC = .CONV$GL_KEY_DESC_BUF;
;  384          0379  2
;  385          0380  2     STUFF_KEY_DESC();
;  386          0381  2
;  387          0382  2     ! Save VBN
;  388          0383  2     !
;  389          0384  2     VBN = .CONV$GL_KEY_DESC_VBN;
;  390          0385  2
;  391          0386  2     ! Write the descriptor we just made
;  392          0387  2     !
;  393          0388  2     CONV$$WRITE_KEY_DESC();
;  394          0389  2
;  395          0390  2     ! Get the key desc just before the one we added
;  396          0391  2     !
;  397          0392  2     CONV$$SET_KEY_DESC( .CONV$GL_ADD_DELE_KEY - 1 );
;  398          0393  2
;  399          0394  2     KEY_DESC [ KEY$L_IDXFL ] = .VBN;
;  400          0395  2
;  401          0396  2     ! Update the key desc making the new one in the chain
;  402          0397  2     !
;  403          0398  2     CONV$$WRITE_KEY_DESC();
;  404          0399  2
;  405          0400  2     RETURN
;  406          0401  2
;  407          0402  1     END;


                                    52  DD 00000 MAKE_DESCRIPTOR:
                                                 PUSHL   R2                                          ; 0313
                              50    0000G CF  D0 00002             MOVL    CONV$AR_AREA_BLOCK, R0     ; 0358
                        50    10    A0    14  A0 C3 00007         SUBL3   20(R0), 16(R0), R0         ; 0359
                                          08  12 0000D             BNEQ    1$                         ; 0358
                                          7E  D4 0000F             CLRL    -(SP)                      ; 0361
                                       0000G  30 00011             BSBW    CONV$$EXTEND_AREA
                                    5E    04  C0 00014             ADDL2   #4, SP                     ;
                                    50    0000G CF  D0 00017 1$:    MOVL    CONV$AR_AREA_BLOCK, R0     ; 0365
                                 0000G CF       18  A0 D0 0001C    MOVL    24(R0), CONV$GL_KEY_DESC_VBN
                                          18    A0  D6 00022       INCL    24(R0)                     ; 0368
                                          14    A0  D6 00025       INCL    20(R0)                     ; 0370
                                          51  D4 00028             CLRL    R1                         ; 0374
                                       0000G  30 0002A             BSBW    CONV$$WRITE_AREA_DESC
                                    5B    0000G CF  D0 0002D       MOVL    CONV$GL_KEY_DESC_BUF, KEY_DESC     ; 0378
                                       0000V  30 00032             BSBW    STUFF_KEY_DESC             ; 0380
                                    52    0000G CF  D0 00035       MOVL    CONV$GL_KEY_DESC_VBN, VBN  ; 0384
                                       0000G  30 0003A             BSBW    CONV$$WRITE_KEY_DESC       ; 0388
```

```
                 7E     0000'  CF          01  C3  0003D          SUBL3   #1, CONV$GL_ADD_DELE_KEY, -(SP)          ; 0392
                                         0000G 30  00043          BSBW    CONV$$SET_KEY_DESC                      ;
                                      5E   04  C0  00046          ADDL2   #4, SP                                  ;
                                      6B   52  D0  00049          MOVL    VBN, (KEY_DESC)                         ; 0394
                                         0000G 30  0004C          BSBW    CONV$$WRITE_KEY_DESC                    ; 0398
                                           04  BA  0004F          POPR    #^M<R2>                                 ; 0402
                                               05  00051          RSB                                             ;

; Routine Size:  82 bytes,    Routine Base:  _CONV$CODE + 00A1
```

```
;  409     0403  1 %SBTTL  'CHECK_XAB'
;  410     0404  1 ROUTINE CHECK_XAB : AL$CHECK_XAB =
;  411     0405  1 !++
;  412     0406  1 !
;  413     0407  1 ! Functional Description:
;  414     0408  1 !
;  415     0409  1 !       Does consistancy checking of the key xab for correctness
;  416     0410  1 !
;  417     0411  1 ! Calling Sequence:
;  418     0412  1 !
;  419     0413  1 !       CHECK_XAB
;  420     0414  1 !
;  421     0415  1 ! Input Parameters:
;  422     0416  1 !       none
;  423     0417  1 !
;  424     0418  1 ! Implicit Inputs:
;  425     0419  1 !
;  426     0420  1 !
;  427     0421  1 ! Output Parameters:
;  428     0422  1 !       none
;  429     0423  1 !
;  430     0424  1 ! Implicit Outputs:
;  431     0425  1 !
;  432     0426  1 !
;  433     0427  1 ! Routine Value:
;  434     0428  1 !
;  435     0429  1 !
;  436     0430  1 ! Routines Called:
;  437     0431  1 !       none
;  438     0432  1 !
;  439     0433  1 ! Side Effects:
;  440     0434  1 !       none
;  441     0435  1 !
;  442     0436  1 !--
;  443     0437  1
;  444     0438  2     BEGIN
;  445     0439  2
;  446     0440  2     DEFINE_XAB;
;  447     0441  2
;  448     0442  2     IF  .XAB [ XAB$B_IAN ] NEQU .XAB [ XAB$B_LAN ]
;  449     0443  2     THEN
;  450     0444  2         RETURN RMS$_IBK;        ! Illegal area
;  451     0445  2
;  452     0446  2     ! Check to see if the areas are in range
;  453     0447  2     !
;  454     0448  2     IF .XAB [ XAB$B_IAN ] GEQU .CONV$AB_OUT_XABSUM [ XAB$B_NOA ]
;  455     0449  2     THEN
;  456     0450  2         RETURN RMS$_IAN;        ! Illegal area
;  457     0451  2
;  458     0452  2     IF .XAB [ XAB$B_LAN ] GEQU .CONV$AB_OUT_XABSUM [ XAB$B_NOA ]
;  459     0453  2     THEN
;  460     0454  2         RETURN RMS$_LAN;        ! Illegal area
;  461     0455  2
;  462     0456  2     IF .XAB [ XAB$B_DAN ] GEQU .CONV$AB_OUT_XABSUM [ XAB$B_NOA ]
;  463     0457  2     THEN
;  464     0458  2         RETURN RMS$_DAN;        ! Illegal area
;  465     0459  2
```

```
;  466     0460  2      ! Stuff some calculated variables
;  467     0461  2      !
;  468     0462  2      XAB [ XAB$B_IBS ] = .CONV$AR_AREA_BLOCK [ .XAB[ XAB$B_IAN ],AREA$B_ARBKTSZ ];
;  469     0463  2      XAB [ XAB$B_DBS ] = .CONV$AR_AREA_BLOCK [ .XAB[ XAB$B_DAN ],AREA$B_ARBKTSZ ];
;  470     0464  2
;  471     0465  2      ! Check data type
;  472     0466  2      !
;  473     0467  2      IF .XAB [ XAB$B_DTP ] GTRU XAB$C_MAXDTP
;  474     0468  2      THEN
;  475     0469  2          RETURN RMS$_DTP;        ! Invalid data type
;  476     0470  2
;  477     0471  2      BEGIN
;  478     0472  2
;  479     0473  2      LOCAL
;  480     0474  2          TOTAL_SIZE      : LONG INITIAL(0);
;  481     0475  2
;  482     0476  2      BIND
;  483     0477  2          KEY_SIZE = XAB [ XAB$B_SIZ0 ] : VECTOR [ ,BYTE ],
;  484     0478  2          KEY_POS  = XAB [ XAB$W_POS0 ] : VECTOR [ ,WORD ];
;  485     0479  2
;  486     0480  2      ! Get the # of segments, total size of the key and the min record size
;  487     0481  2      ! NOTE: NSG and MRL will be zero from FDL$PARSE
;  488     0482  2      !
;  489     0483  2      INCR I FROM 0 TO 7 BY 1
;  490     0484  2      DO
;  491     0485  2
;  492     0486  2          ! Is there a key here
;  493     0487  2          !
;  494     0488  2          IF .KEY_SIZE [ .I ] NEQU 0
;  495     0489  3          THEN
;  496     0490  4              BEGIN
;  497     0491  4
;  498     0492  4              XAB [ XAB$B_NSG ] = .XAB [ XAB$B_NSG ] + 1;
;  499     0493  4
;  500     0494  4              ! Find the total key size
;  501     0495  4              !
;  502     0496  4              IF .KEY_SIZE [ .I ] GTRU .TOTAL_SIZE
;  503     0497  4              THEN
;  504     0498  4                  TOTAL_SIZE = .KEY_SIZE [ .I ];
;  505     0499  4
;  506     0500  4              ! Sind the min. record size
;  507     0501  4              !
;  508     0502  4              IF .XAB [ XAB$W_MRL ] LSSU .KEY_SIZE [ .I ] + .KEY_POS [ .I ]
;  509     0503  4              THEN
;  510     0504  4                  XAB [ XAB$W_MRL ] = .KEY_SIZE [ .I ] + .KEY_POS [ .I ]
;  511     0505  4
;  512     0506  3              END;
;  513     0507  3
;  514     0508  3      ! Check # of segments
;  515     0509  3      !
;  516     0510  3      IF .XAB [ XAB$B_NSG ] EQLU 0
;  517     0511  3      THEN
;  518     0512  3          RETURN 0;           ! Invalid key
;  519     0513  3
;  520     0514  3      ! Check key size
;  521     0515  3      !
;  522     0516  3      IF .TOTAL_SIZE GTRU 255
```

```
;  523     0517  3     THEN
;  524     0518  3         RETURN RMS$_KSZ
;  525     0519  3     ELSE
;  526     0520  3         XAB [ XAB$B_TKS ] = .TOTAL_SIZE
;  527     0521  3
;  528     0522  2     END;
;  529     0523  2
;  530     0524  2     ! Check record size
;  531     0525  2     !
;  532     0526  2     IF ( .CONV$AB_OUT_FAB [ FAB$W_MRS ] NEQU 0 ) AND
;  533     0527  2         ( .CONV$AB_OUT_FAB [ FAB$W_MRS ] LSSU .XAB [ XAB$W_MRL ] )
;  534     0528  2     THEN
;  535     0529  2         RETURN RMS$_POS;        ! Invalid key position
;  536     0530  2
;  537     0531  2     RETURN CONV$_SUCCESS
;  538     0532  2
;  539     0533  1     END;

                                0C  BB 00000 CHECK_XAB:
                                                PUSHR   #^M<R2,R3>                      ; 0404
                    09  AA      08  AA  91 00002        CMPB    8(XAB), 9(XAB)          ; 0442
                                    09  13 00007        BEQL    1$
                        50 0001877C 8F  D0 00009        MOVL    #100220, R0             ; 0444
                                    63  11 00010        BRB     5$
                        50      0000G CF  9A 00012 1$:  MOVZBL  CONV$AB_OUT_XABSUM+8, R0 ; 0448
                        50        08  AA  91 00017        CMPB    8(XAB), R0
                                    09  1F 0001B        BLSSU   2$
                        50 00018554 8F  D0 0001D        MOVL    #99668, R0              ; 0450
                                    4F  11 00024        BRB     5$
                        50        09  AA  91 00026 2$:  CMPB    9(XAB), R0              ; 0452
                                    09  1F 0002A        BLSSU   3$
                        50 000185AC 8F  D0 0002C        MOVL    #99756, R0              ; 0454
                                    40  11 00033        BRB     5$
                        50        0A  AA  91 00035 3$:  CMPB    10(XAB), R0             ; 0456
                                    09  1F 00039        BLSSU   4$
                        50 000184BC 8F  D0 0003B        MOVL    #99516, R0              ; 0458
                                    76  11 00042        BRB     10$
                        50        08  AA  9A 00044 4$:  MOVZBL  8(XAB), R0              ; 0462
    50                  50            06  78 00048        ASHL    #6, R0, R0
                        50      0000G CF  C0 0004C        ADDL2   CONV$AR_AREA_BLOCK, R0
                0C      AA        03  A0  90 00051        MOVB    3(R0), T2(XAB)
                        50        0A  AA  9A 00056        MOVZBL  10(XAB), R0             ; 0463
    50                  50            06  78 0005A        ASHL    #6, R0, R0
                        50      0000G CF  C0 0005E        ADDL2   CONV$AR_AREA_BLOCK, R0
                0D      AA        03  A0  90 00063        MOVB    3(R0), T3(XAB)
                        07        13  AA  91 00068        CMPB    19(XAB), #7             ; 0467
                                    09  1B 0006C        BLEQU   6$
                        50 000184E4 8F  D0 0006E        MOVL    #99556, R0              ; 0469
                                    66  11 00075 5$:    BRB     14$
                                    52  D4 00077 6$:    CLRL    TOTAL_SIZE              ; 0471
                                    50  D4 00079        CLRL    I                       ; 0483
                        51        2E AA40 9A 0007B 7$:  MOVZBL  46(XAB)[I], R1          ; 0488
                                    1F  13 00080        BEQL    9$
```

```
                                    14    AA   96 00082          INCB    20(XAB)                                          ; 0492
                                52          51   D1 00085          CMPL    R1, TOTAL_SIZE                                   ; 0496
                                            03   1B 00088          BLEQU   8$                                               ;
                                52          51   D0 0008A          MOVL    R1, TOTAL_SIZE                                   ; 0498
                                53       1E AA40 3C 0008D 8$:       MOVZWL  30(XAB)[I], R3                                   ; 0502
                                51          53   C0 00092          ADDL2   R3, R1                                           ;
          51        18   AA               10   00   ED 00095          CMPZV   #0, #16, 24(XAB), R1                             ;
                                            04   1E 0009B          BGEQU   9$                                               ;
                                  18    AA          51   B0 0009D          MOVW    R1, 24(XAB)                                      ; 0504
                              D6          50          07   F3 000A1 9$:       AOBLEQ  #7, I, 7$                                ; 0488
                                                14    AA   95 000A5          TSTB    20(XAB)                                  ; 0510
                                                      31   13 000A8          BEQL    13$                                      ;
                                000000FF      8F          52   D1 000AA          CMPL    TOTAL_SIZE, #255                         ; 0516
                                                      09   1B 000B1          BLEQU   11$                                      ;
                                50 000185A4   8F          D0 000B3          MOVL    #99748, R0                               ; 0518
                                            21   11 000BA 10$:       BRB     14$                                      ;
                                  16    AA          52   90 000BC 11$:       MOVB    TOTAL_SIZE, 22(XAB)                      ; 0520
                                50      0000G     CF   3C 000C0          MOVZWL  CONV$AB_OUT_FAB+54, R0                   ; 0526
                                            0F   13 000C5          BEQL    12$                                      ;
                                50        18    AA   B1 000C7          CMPW    24(XAB), R0                              ; 0527
                                            09   1B 000CB          BLEQU   12$                                      ;
                                50 00018624   8F          D0 000CD          MOVL    #99876, R0                               ; 0529
                                            07   11 000D4          BRB     14$                                      ;
                                50          01   D0 000D6 12$:       MOVL    #1, R0                                   ; 0531
                                            02   11 000D9          BRB     14$                                      ;
                                            50   D4 000DB 13$:       CLRL    R0                                       ; 0533
                                            0C   BA 000DD 14$:       POPR    #^M<R2,R3>                               ;
                                                 05 000DF          RSB                                                      ;

; Routine Size:  224 bytes,    Routine Base:  _CONV$CODE + 00F3
```

```
;   541          0534  1 %SBTTL  'STUFF_KEY_DESC'
;   542          0535  1 ROUTINE STUFF_KEY_DESC : AL$STUFF_KEY_DESC NOVALUE =
;   543          0536  1 !++
;   544          0537  1 !
;   545          0538  1 ! Functional Description:
;   546          0539  1 !
;   547          0540  1 !       Does the stuffing of the index descriptor from the key xab
;   548          0541  1 !
;   549          0542  1 ! Calling Sequence:
;   550          0543  1 !
;   551          0544  1 !       STUFF_KEY_DESC()
;   552          0545  1 !
;   553          0546  1 ! Input Parameters:
;   554          0547  1 !       none
;   555          0548  1 !
;   556          0549  1 ! Implicit Inputs:
;   557          0550  1 !
;   558          0551  1 !
;   559          0552  1 ! Output Parameters:
;   560          0553  1 !       none
;   561          0554  1 !
;   562          0555  1 ! Implicit Outputs:
;   563          0556  1 !
;   564          0557  1 !
;   565          0558  1 ! Routine Value:
;   566          0559  1 !       none
;   567          0560  1 !
;   568          0561  1 ! Routines Called:
;   569          0562  1 !       none
;   570          0563  1 !
;   571          0564  1 ! Side Effects:
;   572          0565  1 !       none
;   573          0566  1 !
;   574          0567  1 !--
;   575          0568  1
;   576          0569  2     BEGIN
;   577          0570  2
;   578          0571  2     DEFINE_XAB;
;   579          0572  2     DEFINE_KEY_DESC;
;   580          0573  2
;   581          0574  2     ! Zero the descriptor
;   582          0575  2     !
;   583          0576  2     CH$FILL( 0,KEY$C_BLN,.KEY_DESC );
;   584          0577  2
;   585          0578  2     ! Copy the xab into the descriptor (this cheat should always work)
;   586          0579  2     !
;   587          0580  2     CH$MOVE( XAB$C_KEYLEN - 4,XAB [ XAB$B_IAN ],KEY_DESC [ KEY$B_IANUM ] );
;   588          0581  2
;   589          0582  2     ! Copy key name if any
;   590          0583  2     !
;   591          0584  2     IF .XAB [ XAB$L_KNM ] NEQU 0
;   592          0585  2     THEN
;   593          0586  2         CH$MOVE( 32,.XAB [ XAB$L_KNM ],KEY_DESC [ KEY$T_KEYNAM ] );
;   594          0587  2
;   595          0588  2     ! Stuff the data type into seq0 type (prologue 3 screwup) and zero the rest
;   596          0589  2     !
;   597          0590  2     KEY_DESC [ KEY$B_TYPE0 ] = .KEY_DESC [ KEY$B_DATATYPE ];
```

```
;  598          0591  2
;  599          0592  2      CH$FILL( 0,7,KEY_DESC [ KEY$B_TYPE1 ] );
;  600          0593  2
;  601          0594  2      ! Finally say that the index is not initialized
;  602          0595  2      !
;  603          0596  2      KEY_DESC [ KEY$V_INITIDX ] = _SET;
;  604          0597  2
;  605          0598  2      RETURN
;  606          0599  2
;  607          0600  1      END;


                                                3C  BB 00000 STUFF_KEY_DESC:
                                                             PUSHR   #^M<R2,R3,R4,R5>                        ; 0535
    0060  8F          00          6E                00  2C 00002          MOVC5   #0, (SP), #0, #96, (KEY_DESC)   ; 0576
                                                    6B    00009
                  06  AB      08  AA      0048      8F  28 0000A          MOVC3   #72, 8(XAB), 6(KEY_DESC)        ; 0580
                                            38      AA  D5 00012          TSTL    56(XAB)                         ; 0584
                                                    06  13 00015          BEQL    1$                              ;
                  34  AB      38  BA                20  28 00017          MOVC3   #32, @56(XAB), 52(KEY_DESC)     ; 0586
                              58  AB          11      AB  90 0001D 1$:      MOVB    17(KEY_DESC), 88(KEY_DESC)      ; 0590
          07                  00          6E            00  2C 00022      MOVC5   #0, (SP), #0, #7, 89(KEY_DESC)  ; 0592
                                                59      AB    00027
                              10  AB                    10  88 00029      BISB2   #16, 16(KEY_DESC)               ; 0596
                                                        3C  BA 0002D      POPR    #^M<R2,R3,R4,R5>                ; 0600
                                                            05 0002F      RSB                                     ;

; Routine Size:  48 bytes,    Routine Base:  _CONV$CODE + 01D3
```

```
;  609     0601  1 %SBTTL  'LOAD_KEY'
;  610     0602  1 GLOBAL ROUTINE  ADD$$LOAD_KEY : AL$LOAD_KEY =
;  611     0603  1 !++
;  612     0604  1 !
;  613     0605  1 ! Functional Description:
;  614     0606  1 !
;  615     0607  1 !
;  616     0608  1 ! Calling Sequence:
;  617     0609  1 !
;  618     0610  1 !
;  619     0611  1 ! Input Parameters:
;  620     0612  1 !       none
;  621     0613  1 !
;  622     0614  1 ! Implicit Inputs:
;  623     0615  1 !
;  624     0616  1 !
;  625     0617  1 ! Output Parameters:
;  626     0618  1 !       none
;  627     0619  1 !
;  628     0620  1 ! Implicit Outputs:
;  629     0621  1 !
;  630     0622  1 !
;  631     0623  1 ! Routine Value:
;  632     0624  1 !
;  633     0625  1 !
;  634     0626  1 ! Routines Called:
;  635     0627  1 !
;  636     0628  1 !
;  637     0629  1 ! Side Effects:
;  638     0630  1 !       none
;  639     0631  1 !
;  640     0632  1 !--
;  641     0633  1
;  642     0634  2     BEGIN
;  643     0635  2
;  644     0636  2     DEFINE_KEY_DESC;
;  645     0637  2     DEFINE_CTX_GLOBAL;
;  646     0638  2     DEFINE_BUCRET_GLOBAL;
;  647     0639  2
;  648     0640  2     LOCAL
;  649     0641  2         STATUS;
;  650     0642  2
;  651     0643  2     ! Get the new key
;  652     0644  2     !
;  653     0645  2     CONV$$SET_KEY_DESC( .CONV$GL_ADD_DELE_KEY );
;  654     0646  2
;  655     0647  2     ! Init the fast load process
;  656     0648  2     !
;  657     0649  2     CONV$$INIT_FAST_LOAD( .KEY_DESC [ KEY$B_KEYSZ ] );
;  658     0650  2
;  659     0651  2     CONV$GL_WORK_F = 2;
;  660     0652  2
;  661     0653  2     ! Set up the sort for the secondary key.  The sort is a INDEX sort.
;  662     0654  2     ! This type of sort will produce a file of RFA's and keys of the
;  663     0655  2     ! primary data level we just made.
;  664     0656  2     !
;  665     0657  3     IF NOT ( STATUS = CONV$$SORT_SECONDARY() )
```

```
;  666          0658  2      THEN
;  667          0659  2          RETURN .STATUS;
;  668          0660  2
;  669          0661  2      ! Now that the file is sorted get the data and load it in.
;  670          0662  2      !
;  671          0663  2      CONV$$LOAD_SECONDARY();
;  672          0664  2
;  673          0665  2      ! Write the prologue (area desc)
;  674          0666  2      !
;  675          0667  2      CONV$$WRITE_PROLOGUE();
;  676          0668  2
;  677          0669  2      ! And the key descriptor
;  678          0670  2      !
;  679          0671  2      CONV$$WRITE_KEY_DESC();
;  680          0672  2
;  681          0673  2      RETURN 1
;  682          0674  2
;  683          0675  1      END;


                        7E            59  7D 00000 ADD$$LOAD_KEY::
                                                       MOVQ    R9, -(SP)                                    ;  0602
                      0000'   CF      DD 00003         PUSHL   CONV$GL_ADD_DELE_KEY                         ;  0645
                         0000G 30 00007                BSBW    CONV$$SET_KEY_DESC                           ;
                        6E      14    AB  9A 0000A     MOVZBL  20(KEY_DESC), (SP)                           ;  0649
                         0000G 30 0000E                BSBW    CONV$$INIT_FAST_LOAD                         ;
                        5E            04  C0 00011     ADDL2   #4, SP                                       ;
                  0000G CF            02  D0 00014     MOVL    #2, CONV$GL_WORK_F                           ;  0651
                         0000G 30 00019                BSBW    CONV$$SORT_SECONDARY                         ;  0657
                        0E            50  E9 0001C     BLBC    STATUS, 1$                                   ;
                         0000G 30 0001F                BSBW    CONV$$LOAD_SECONDARY                         ;  0663
                  0000G CF            00  FB 00022     CALLS   #0, CONV$$WRITE_PROLOGUE                     ;  0667
                         0000G 30 00027                BSBW    CONV$$WRITE_KEY_DESC                         ;  0671
                        50            01  D0 0002A     MOVL    #1, R0                                       ;  0673
                        59            8E  7D 0002D 1$:  MOVQ    (SP)+, R9                                    ;  0675
                                      05 00030          RSB                                                  ;

; Routine Size:  49 bytes,    Routine Base:  _CONV$CODE + 0203

;  684          0676  1
;  685          0677  0 END     ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _CONV$GLOBAL                    4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _CONV$CODE                    564  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]LIB.L32;1          18619         63         0        1000       00:01.8
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1      165         19        11          17       00:00.2



;                                       COMMAND QUALIFIERS

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

; Size:          564 code + 4 data bytes
; Run Time:        00:14.9
; Elapsed Time:    00:50.2
; Lines/CPU Min:    2726
; Lexemes/CPU-Min: 16167
; Memory Used:  139 pages
; Compilation Complete
```

COBRMSBLO
LIS

COBSPANC
LIS

COBSWITCH
LIS

COBRESTVA
LIS

CONVSHR
MAP

CONVCOMIO
LIS

COBSETSWI
LIS

CONVDEF
MDL

CONVLINK
REQ

COBSPANC2
LIS

CONV

CONVERT
MAP

CONVCALL
LIS

COBVECTOR
LIS

CONVERT
REQ

RECLAIM
MAP

ADDKEY
LIS