

CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV
CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV
CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCC	000	000	NNN	VVV	VVV
CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV
CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV
CCCCCCCCCCCC	00000000	NNN	NNN	VVV	VVV

```

CCCCCCCC 000000 NN NN VV VV FFFFFFFF IIIIII LL EEEEEEEEE SSSSSSS
CCCCCCCC 000000 NN NN VV VV FFFFFFFF IIIIII LL EEEEEEEEE SSSSSSS
CC 00 00 NN NN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NN NN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NNNN NN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NNNN NN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NN NN NN VV VV FFFFFFFF IIII LL EEEEEEEEE SSSSSS
CC 00 00 NN NN NN VV VV FFFFFFFF IIII LL EEEEEEEEE SSSSSS
CC 00 00 NN NNNN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NN NNNN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NN NN VV VV FF IIII LL EEEEEEEEE SS
CC 00 00 NN NN VV VV FF IIII LL EEEEEEEEE SS
CCCCCCCC 000000 NN NN VV VV FFFFFFFF IIIIII LLLLLLLLLL EEEEEEEEE SSSSSSS
CCCCCCCC 000000 NN NN VV VV FFFFFFFF IIIIII LLLLLLLLLL EEEEEEEEE SSSSSSS

```

```

LL IIIIII SSSSSSS
LL IIIIII SSSSSSS
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 SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

```

0001 0 %TITLE 'VAX-11 CONVERT'
0002 0 MODULE CONV$FILES ( IDENT='V04-000',
0003 0 OPTLEVEL=3
0004 0 ) =
0005 0
0006 1 BEGIN
0007 1
0008 1 :*****
0009 1 :*
0010 1 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 :* ALL RIGHTS RESERVED.
0013 1 :*
0014 1 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 :* TRANSFERRED.
0020 1 :*
0021 1 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 :* CORPORATION.
0024 1 :*
0025 1 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 :*
0028 1 :*
0029 1 :*****

```

```

31 0030 1  **
32 0031 1
33 0032 1 Facility: VAX-11 CONVERT
34 0033 1
35 0034 1 Abstract: RMS file handling routines
36 0035 1
37 0036 1 Contents:
38 0037 1 PARSE_DEF
39 0038 1 OPEN_INPUT
40 0039 1 SEARCH_FILE
41 0040 1 OPEN_IN
42 0041 1 OPEN_OUTPUT
43 0042 1 GET_PROLOGUE
44 0043 1 CREATE_BUFFER
45 0044 1
46 0045 1 Environment:
47 0046 1
48 0047 1 VAX/VMS Operating System
49 0048 1
50 0049 1
51 0050 1 Author: Keith B. Thompson Creation date: June-1980
52 0051 1
53 0052 1
54 0053 1 Modified by:
55 0054 1
56 0055 1 V03-013 JWT0194 Jim Teague 31-Aug-1984
57 0056 1 Fix problem with CONVERT dropping blocks when input
58 0057 1 file is UDF.
59 0058 1
60 0059 1 V03-012 RAS0311 Ron Schaefer 18-Jun-1984
61 0060 1 Fix output file related file parsing by making sure the
62 0061 1 input file result filespec is available. Fix to RAS0260.
63 0062 1
64 0063 1 V03-011 RAS0272 Ron Schaefer 16-Mar-1984
65 0064 1 Allow CONVERT to fastload & sort network files since
66 0065 1 SORT-32 is now able to handle them.
67 0066 1
68 0067 1 V03-010 RAS0260 Ron Schaefer 2-Mar-1984
69 0068 1 Improve performance of RAS0250 by copying the DVI, FID and
70 0069 1 DID fields from the LIBSFIND_FILE NAM to the real NAM
71 0070 1 used for the open. Also copy the device characteristics.
72 0071 1
73 0072 1 V03-009 RAS0250 Ron Schaefer 23-Feb-1984
74 0073 1 Convert SEARCH_FILE to use LIBSFIND_FILE for correct
75 0074 1 related file processing. Add FDL_STRING support.
76 0075 1
77 0076 1 V03-008 KBT0442 Keith B. Thompson 30-Dec-1982
78 0077 1 Make fdl_fab/rab global
79 0078 1
80 0079 1 V03-007 KBT0435 Keith B. Thompson 16-Dec-1982
81 0080 1 Always open the input file to fill the fab
82 0081 1 except when coming from tape and sorting
83 0082 1
84 0083 1 V03-006 KBT0392 Keith B. Thompson 29-Oct-1982
85 0084 1 Call new read_prologue routine
86 0085 1
87 0086 1 V03-005 KBT0370 Keith B. Thompson 19-Oct-1982

```

:	88	0087	1	:		Use new supported fdlsparse
:	89	0088	1	:		
:	90	0089	1	:	V03-004	KBT0347 Keith B. Thompson 4-Oct-1982
:	91	0090	1	:		Use new Linkage definitions
:	92	0091	1	:		
:	93	0092	1	:	V03-003	KBT0044 Keith Thompson 5-Apr-1982
:	94	0093	1	:		Don't do a search on a device mounted foreign
:	95	0094	1	:		
:	96	0095	1	:	V03-002	KBT0025 Keith Thompson 26-Mar-1982
:	97	0096	1	:		Fix fill switch for /nofast
:	98	0097	1	:		
:	99	0098	1	:	V03-001	KBT0015 Keith Thompson 18-Mar-1982
:	100	0099	1	:		Fix area allocation bug in get_prologue and use new
:	101	0100	1	:		plgSc_ver3 instead of literal
:	102	0101	1	:		
:	103	0102	1	:		****

```

105 0103 1
106 0104 1 PSECT
107 0105 1      OWN      = CONV$OWN      (PIC),
108 0106 1      GLOBAL  = CONV$GLOBAL (PIC),
109 0107 1      PLIT    = CONV$PLIT   (SHARE,PIC),
110 0108 1      CODE    = CONV$CODE  (SHARE,PIC);
111 0109 1
112 0110 1 LIBRARY 'SYS$LIBRARY:LIB.L32';
113 0111 1 LIBRARY 'SRCS:CONVERT';
114 0112 1
115 0113 1 EXTERNAL ROUTINE
116 0114 1      CONV$$GET_VM : CL$GET_VM,
117 0115 1      CONV$$READ_PROLOGUE : CL$READ_PROLOGUE NOVALUE,
118 0116 1      CONV$$RMS_OPEN_ERROR : NOVALUE,
119 0117 1      FDL$PARSE : ADDRESSING_MODE( GENERAL ),
120 0118 1      LIB$FIND_FILE : ADDRESSING_MODE( GENERAL );
121 0119 1
122 0120 1 FORWARD ROUTINE
123 0121 1      CONV$$SEARCH_FILE,
124 0122 1      CONV$$OPEN_IN;
125 0123 1
126 0124 1 : Error codes
127 0125 1 :
128 0126 1 DEFINE_ERROR_CODES;
129 0127 1
130 0128 1 EXTERNAL
131 0129 1
132 0130 1 : The Option Flags:
133 0131 1 :
134 0132 1      CONV$GL_APPEND : LONG, : APPEND
135 0133 1      CONV$GL_CREATE : LONG, : CREATE
136 0134 1      CONV$GL_FDL : LONG, : FDL
137 0135 1      CONV$GL_EXC : LONG, : EXCEPTION
138 0136 1      CONV$GL_FAST : LONG, : FAST
139 0137 1      CONV$GL_MERGE : LONG, : MERGE
140 0138 1      CONV$GL_FILL : LONG, : FILL_BUCKETS
141 0139 1      CONV$GL_FIX : LONG, : FIXED_WRITE
142 0140 1      CONV$GL_KEY : LONG, : KEY
143 0141 1      CONV$GL_PAD : LONG, : PAD_RECORDS
144 0142 1      CONV$GL_SHARE : LONG, : SHARE
145 0143 1      CONV$GL_SORT : LONG, : SORT
146 0144 1      CONV$GL_READ_C : LONG, : READ_CHECK
147 0145 1      CONV$GL_TRUNCATE : LONG, : TRUNCATE
148 0146 1      CONV$GL_WRITE_C : LONG, : WRITE_CHECK
149 0147 1      CONV$GL_PROLOG : LONG, : PROLOGUE
150 0148 1      CONV$AB_FLAGS : BLOCK [ ,BYTE ],
151 0149 1
152 0150 1      CONV$GW_OUT_MRS : WORD,
153 0151 1      CONV$GW_UDF_MRS : WORD,
154 0152 1      CONV$GB_CURRENT_FILE : BYTE,
155 0153 1      CONV$GW_MAX_REC_SIZ : WORD,
156 0154 1      CONV$GL_REC_BUF_PTR,
157 0155 1      CONV$GL_VFC_BUF_PTR,
158 0156 1      CONV$GL_FINDFILE_CTX,
159 0157 1
160 0158 1      CONV$AL_IN_FILE_NAM : VECTOR [ ,LONG ], : Input File
161 0159 1      CONV$AR_OUT_FILE_NAM : REF_DESC_BLK, : Output File

```

:	162	0160	1	CONVSAR_FDL_FILE_NAM	:	REF DESC_BLK,	! FDL File
:	163	0161	1				
:	164	0162	1	CONVSAB_IN_XABSUM	:	\$XABSUM_DECL,	
:	165	0163	1	CONVSAB_IN_XABFHC	:	\$XABFHC_DECL,	
:	166	0164	1	CONVSAB_IN_NAM	:	\$NAM_DECL,	
:	167	0165	1	CONVSAB_IN_FAB	:	\$FAB_DECL,	
:	168	0166	1	CONVSAB_IN_RAB	:	\$RAB_DECL,	
:	169	0167	1	CONVSAB_OUT_XABSUM	:	\$XABSUM_DECL,	
:	170	0168	1	CONVSAB_OUT_NAM	:	\$NAM_DECL,	
:	171	0169	1	CONVSAB_OUT_FAB	:	\$FAB_DECL,	
:	172	0170	1	CONVSAB_OUT_RAB	:	\$RAB_DECL;	
:	173	0171	1				
:	174	0172	1	GLOBAL			
:	175	0173	1	CONVSAB_FDL_FAB	:	REF BLOCK [,BYTE],	
:	176	0174	1	CONVSAB_FDL_RAB	:	REF BLOCK [,BYTE];	
:	177	0175	1				

```

179 0176 1 %SBTTL 'PARSE_DEF'
180 0177 1 GLOBAL ROUTINE=CONV$$PARSE_DEF =
181 0178 1 ++
182 0179 1
183 0180 1 Functional Description:
184 0181 1
185 0182 1     Calls fdlsparse to parse the fdl file and fill in a fab. The
186 0183 1     info from this fab 's will be copied to the output fab in open_output
187 0184 1
188 0185 1 Calling Sequence:
189 0186 1
190 0187 1     CONV$$PARSE()
191 0188 1
192 0189 1 Input Parameters:
193 0190 1     none
194 0191 1
195 0192 1 Implicit Inputs:
196 0193 1
197 0194 1     CONV$AR_FDL_FILE_NAME - FDL file descriptor
198 0195 1
199 0196 1 Output Parameters:
200 0197 1     none
201 0198 1
202 0199 1 Implicit Outputs:
203 0200 1     none
204 0201 1
205 0202 1 Routine Value:
206 0203 1
207 0204 1     Value returned by fdlsparse
208 0205 1
209 0206 1 Routines Called:
210 0207 1
211 0208 1     FDL$PARSE
212 0209 1
213 0210 1 Side Effects:
214 0211 1
215 0212 1 --
216 0213 1
217 0214 2 BEGIN
218 0215 2
219 0216 2 ! EXTERNAL LITERAL
220 0217 2 !     FDL$M_FDL_STRING,
221 0218 2 !     FDL$M_SIGNAL;
222 0219 2
223 0220 2 LOCAL
224 0221 2 !     FDL_FLAGS      : LONG;
225 0222 2
226 0223 2 ! Initialize the flags
227 0224 2
228 0225 2 FDL_FLAGS = 0;
229 0226 2
230 0227 2 ! If convert is signaling then fdl should
231 0228 2 !
232 0229 2 IF .CONV$AB_FLAGS [ CONV$V_SIGNAL ]
233 0230 2 THEN
234 0231 2 !     FDL_FLAGS = FDL$M_SIGNAL;
235 0232 2 !     FDL_FLAGS = 1;

```



```

: 236
: 237
: 238
: 239
: 240
: 241
: 242
: 243
: 244
: 245
: 246
: 247
: 248
: 249
: 250
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247

```

```

: If caller passed in an fdl string, then tell fdl about it
IF .CONVSAB_FLAGS [ CONVS_FDL_STRING ]
THEN
  FDL_FLAGS = FDL$M_FDL_STRING OR .FDL_FLAGS;
  FDL_FLAGS = 2 OR .FDL_FLAGS;

RETURN FDL$PARSE( .CONVSAB_FDL_FILE_NAM,
                  CONVSAB_FDL_FAB,
                  CONVSAB_FDL_RAB,
                  FDL_FLAGS )

END;

```

```

.TITLE CONVFILES VAX-11 CONVERT
.IDENT \V04-000\

.PSECT _CONV$GLOBAL,NOEXE, PIC,2

00000 CONVSAB_FDL_FAB::
      .BLKB 4
00004 CONVSAB_FDL_RAB::
      .BLKB 4

.EXTRN CONV$GET_VM, CONV$READ_PROLOGUE
.EXTRN CONV$RMS_OPEN_ERROR
.EXTRN FDL$PARSE, LIB$FIND_FILE
.EXTRN CONVERT$FACILITY
.EXTRN CONV$FAD_MAX, CONV$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$WRITEERR
.EXTRN CONV$GL_APPEND, CONV$GL_CREATE
.EXTRN CONV$GL_FDL, CONV$GL_EXC

```

```

.EXTRN CONV$GL_FAST, CONV$GL_MERGE
.EXTRN CONV$GL_FILL, CONV$GL_FIX
.EXTRN CONV$GL_KEY, CONV$GL_PAD
.EXTRN CONV$GL_SHARE, CONV$GL_SORT
.EXTRN CONV$GL_READ_C, CONV$GL_TRUNCATE
.EXTRN CONV$GL_WRITE_C
.EXTRN CONV$GL_PROLOG, CONV$AB_FLAGS
.EXTRN CONV$GW_OUT_MRS
.EXTRN CONV$GW_UDF_MRS
.EXTRN CONV$GB_CURRENT_FILE
.EXTRN CONV$GW_MAX_REC_SIZ
.EXTRN CONV$GL_REC_BUF_PTR
.EXTRN CONV$GL_VFC_BUF_PTR
.EXTRN CONV$GL_FINDFILE_CTX
.EXTRN CONV$AR_IN_FILE_NAM
.EXTRN CONV$AR_OUT_FILE_NAM
.EXTRN CONV$AR_FDL_FILE_NAM
.EXTRN CONV$AB_IN_XABSUM
.EXTRN CONV$AB_IN_XABFHC
.EXTRN CONV$AB_IN_NAM, CONV$AB_IN_FAB
.EXTRN CONV$AB_IN_RAB, CONV$AB_OUT_XABSUM
.EXTRN CONV$AB_OUT_NAM
.EXTRN CONV$AB_OUT_FAB
.EXTRN CONV$AB_OUT_RAB

```

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

```

.ENTRY CONV$$PARSE_DEF, Save nothing : 0177
CLR  FDL_FLAGS : 0225
BLBC CONV$AB_FLAGS, 1$ : 0229
MOVL #1, FDL_FLAGS : 0232
BBC #1, CONV$AB_FLAGS, 2$ : 0236
BISB2 #2, FDL_FLAGS : 0239
PUSHL SP : 0242
PUSHAB CONV$AB_FDL_RAB
PUSHAB CONV$AB_FDL_FAB
PUSHL CONV$AR_FDL_FILE_NAM
CALLS #4, FDL$PARSE : 0247
RET

```

```

0000 00000
7E D4 00002
03 0000G CF E9 00004
6E 01 D0 00009
03 0000G CF 01 E1 0000C 1$:
6E 02 88 00012
SE DD 00015 2$:
0000' CF 9F 00017
0000' CF 9F 0001B
0000G CF DD 0001F
00000000G 00 04 FB 00023
04 0002A

```

: Routine Size: 43 bytes, Routine Base: _CONV\$CODE + 0000

```

: 252      0248 1 %SBTTL 'OPEN INPUT'
: 253      0249 1 GLOBAL ROUTINE CONV$$OPEN_INPUT =
: 254      0250 1 ++
: 255      0251 1
: 256      0252 1 : Functional Description:
: 257      0253 1
: 258      0254 1 :     Opens an input file
: 259      0255 1
: 260      0256 1 : Calling Sequence:
: 261      0257 1
: 262      0258 1 :     CONV$$OPEN_INPUT()
: 263      0259 1
: 264      0260 1 : Input Parameters:
: 265      0261 1 :     none
: 266      0262 1
: 267      0263 1 : Implicit Inputs:
: 268      0264 1 :     none
: 269      0265 1
: 270      0266 1 : Output Parameters:
: 271      0267 1 :     none
: 272      0268 1
: 273      0269 1 : Implicit Outputs:
: 274      0270 1 :     none
: 275      0271 1
: 276      0272 1 : Routine Value:
: 277      0273 1
: 278      0274 1 :     CONV$_SUCCESS or error code from CONV$$SEARCH_FILE or CONV$$OPEN_IN
: 279      0275 1
: 280      0276 1 : Routines Called:
: 281      0277 1
: 282      0278 1 :     CONV$$SEARCH_FILE
: 283      0279 1 :     CONV$$OPEN_IN
: 284      0280 1
: 285      0281 1 : Side Effects:
: 286      0282 1
: 287      0283 1 :     Opens an input file
: 288      0284 1
: 289      0285 1 : --
: 290      0286 1
: 291      0287 2 BEGIN
: 292      0288 2
: 293      0289 2 LOCAL
: 294      0290 2     STATUS  : LONG,
: 295      0291 2     IN_DEV  : BLOCK [ 1, LONG ];
: 296      0292 2
: 297      0293 2
: 298      0294 2 : Any errors on the input fab are OPENIN errors
: 299      0295 2
: 300      0296 2 CONV$AB_IN_FAB [ FAB$$_CTX ] = CONV$_OPENIN;
: 301      0297 2
: 302      0298 2 : Start by getting the file name
: 303      0299 2
: 304      0300 2 RET_ON_ERROR( CONV$$SEARCH_FILE() );
: 305      0301 2
: 306      0302 2 : For now there are only sequential files on tape if there is no
: 307      0303 2 : definition file then it cant be a fast load
: 308      0304 2

```

```

: 309      0305      2      IN_DEV = .CONV$AB_IN_FAB [ FAB$L_DEV ];
: 310      0306      2
: 311      0307      2      IF ( NOT .CONV$GL_FDL ) AND .IN_DEV [ DEV$V_SQD ]
: 312      0308      2      THEN
: 313      0309      2          BEGIN
: 314      0310      2              CONV$GL_FAST      = _CLEAR;
: 315      0311      2              CONV$GL_SORT      = _CLEAR;
: 316      0312      2          END;
: 317      0313      2
: 318      0314      2      : If we are sorting the input file from tape or network
: 319      0315      2      : then don't bother opening the input file here
: 320      0316      2
: 321      0317      2      IF .CONV$GL_SORT AND (.IN_DEV [ DEV$V_SQD ] OR .IN_DEV [ DEV$V_NET ])
: 322      0318      2      THEN
: 323      0319      2          RETURN CONV$_SUCCESS
: 324      0320      2      ELSE
: 325      0321      2          RETURN CONV$$OPEN_IN()
: 326      0322      2
: 327      0323      1      END;

```

			0000	00000	.ENTRY	CONV\$\$OPEN INPUT, Save nothing	: 0249
	0000G	CF	00000000G	8F D0 00002	MOVL	#CONV\$ OPENIN, CONV\$AB_IN_FAB+24	: 0296
	0000V	CF		00 FB 0000B	CALLS	#0, CONV\$\$SEARCH_FILE	: 0300
		2C		50 E9 00010	BLBC	STATUS, 4\$	
		50	0000G	CF D0 00013	MOVL	CONV\$AB_IN FAB+64, IN_DEV	: 0305
08		0C	0000G	CF E8 00018	BLBS	CONV\$GL_FDL, 1\$: 0307
		50		05 E1 0001D	BBC	#5, IN_DEV, 1\$	
			0000G	CF D4 00021	CLRL	CONV\$GL_FAST	: 0310
			0000G	CF D4 00025	CLRL	CONV\$GL_SORT	: 0311
04		0C	0000G	CF E9 00029 1\$:	BLBC	CONV\$GL_SORT, 3\$: 0317
04		50		05 E0 0002E	BBS	#5, IN_DEV, 2\$	
		50		0D E1 00032	BBC	#13, IN_DEV, 3\$	
		50		01 D0 00036 2\$:	MOVL	#1, R0	: 0321
				04 00039	RET		
	0000V	CF		00 FB 0003A 3\$:	CALLS	#0, CONV\$\$OPEN_IN	: 0323
				04 0003F 4\$:	RET		

; Routine Size: 64 bytes, Routine Base: _CONV\$CODE + 002B

```

329 0324 1 %SBTTL 'SEARCH_FILE'
330 0325 1 GLOBAL ROUTINE CONV$$SEARCH_FILE =
331 0326 1 **
332 0327 1
333 0328 1 Functional Description:
334 0329 1
335 0330 1     Searches for an input file
336 0331 1
337 0332 1 Calling Sequence:
338 0333 1
339 0334 1     CONV$$SEARCH_FILE()
340 0335 1
341 0336 1 Input Parameters:
342 0337 1     none
343 0338 1
344 0339 1 Implicit Inputs:
345 0340 1
346 0341 1     CONV$AL_IN_FILE_NAME     - Input file name descriptor array
347 0342 1     CONV$GB_CURRENT_FILE     - Current input file name being searched
348 0343 1     CONV$GL_FINDFILE_CTX     - LIB$FIND_FILE context
349 0344 1
350 0345 1 Output Parameters:
351 0346 1     none
352 0347 1
353 0348 1 Implicit Outputs:
354 0349 1     none
355 0350 1
356 0351 1 Routine Value:
357 0352 1
358 0353 1     CONV$_SUCCESS or CONV$_NOWILD
359 0354 1
360 0355 1 Routines Called:
361 0356 1
362 0357 1     LIB$FIND_FILE
363 0358 1     $PARSE
364 0359 1     CONV$$RMS_OPEN_ERROR     - By RMS as an AST or by us on LIB$FIND_FILE errors
365 0360 1     $SEARCH
366 0361 1
367 0362 1 Side Effects:
368 0363 1
369 0364 1     Sets up input name block for next input file
370 0365 1
371 0366 1 --
372 0367 1
373 0368 2 BEGIN
374 0369 2
375 0370 2 LOCAL
376 0371 2     STATUS,
377 0372 2     STV,
378 0373 2     FINDFILENAM     : REF BLOCK [ ,BYTE ],
379 0374 2     IN_NAME         : REF DESC_BLK,
380 0375 2     OUT_NAME        : DESC_BLK,
381 0376 2     IN_DEVICE       : BLOCK [ 1, LONG ];
382 0377 2
383 0378 2 BIND
384 0379 2     FINDFILEFAB     = CONV$GL_FINDFILE_CTX : REF BLOCK[ ,BYTE];
385 0380 2

```

```

386 0381 2 IN_NAME = .CONVSAL_IN_FILE_NAM [ .CONV$GB_CURRENT_FILE ];
387 0382 2
388 0383 2 OUT_NAME [DSC$B_CLASS] = DSC$K_CLASS_D;
389 0384 2 OUT_NAME [DSC$B_DTYPE] = DSC$K_DTYPE_T;
390 0385 2 OUT_NAME [DSC$W_LENGTH] = 0;
391 0386 2 OUT_NAME [DSC$A_POINTER] = 0;
392 0387 2
393 0388 2 ! Get the next file name to search for
394 0389 2
395 0390 2 STATUS = LIB$FIND_FILE(
396 0391 2     .IN_NAME, OUT_NAME,
397 0392 2     CONV$GL_FINDFILE_CTX,
398 0393 2     0, 0, STV, %REF(3));
399 0394 2
400 0395 2 ! If the filename has wildcards in it it's an error
401 0396 2
402 0397 2 IF (.STATUS AND ST$M_MSG_NO) EQL SHR$NOWILD
403 0398 2 THEN
404 0399 2     RETURN CONV$NOWILD;
405 0400 2
406 0401 2 ! Report miscellaneous errors from LIB$FIND_FILE
407 0402 2
408 0403 2 IF NOT .STATUS
409 0404 2 THEN
410 0405 2     BEGIN
411 0406 2     FINDFILEFAB [ FAB$S_CTX ] = CONV$ OPENIN;
412 0407 2     CONV$$RMS_OPEN_ERROR(.FINDFILEFAB);
413 0408 2     END;
414 0409 2
415 0410 2 CONV$AB_IN_FAB [ FAB$B_FNS ] = .OUT_NAME [ DSC$W_LENGTH ];
416 0411 2 CONV$AB_IN_FAB [ FAB$S_FNA ] = .OUT_NAME [ DSC$A_POINTER ];
417 0412 2
418 0413 2 ! Clear the IFI and device char. so we can parse
419 0414 2
420 0415 2 CONV$AB_IN_FAB [ FAB$W_IFI ] = 0;
421 0416 2 CONV$AB_IN_FAB [ FAB$S_DEV ] = .FINDFILEFAB [ FAB$S_DEV ];
422 0417 2
423 0418 2 FINDFILENAM = .FINDFILEFAB [ FAB$S_NAM ];
424 0419 2
425 0420 2 ! Copy the DVI, FID and DID fields to the NAM block to use for the open.
426 0421 2
427 0422 2 CH$MOVE( NAM$S_DVI+NAM$S_FID+NAM$S_DID,
428 0423 2     FINDFILENAM [ NAM$T_DVI ], CONV$AB_IN_NAM [ NAM$T_DVI ] );
429 0424 2
430 0425 2 RETURN CONV$SUCCESS
431 0426 2
432 0427 1 END;

```

```

007C 0000
56 0000G CF 9E 00002
5E 00 0C C2 00007
50 0000G CF 9A 0000A
50 0000GCF40 D0 0000F

```

```

.ENTRY CONV$$SEARCH_FILE, Save R2,R3,R4,R5,R6
MOVAB CONV$GL_FINDFILE_CTX, R6
SUBL2 #12, SP
MOVZBL CONV$GB_CURRENT_FILE, R0
MOVL CONV$AL_IN_FILE_NAM[R0], IN_NAME

```

```

: 0325
:
: 0381
:

```

04	AE	020E0000	8F	D0	00015	MOVL	#34471936, OUT_NAME	:	0385	
		08	AE	D4	0001D	CLRL	OUT_NAME+2	:	0386	
			03	DD	00020	PUSHL	#3	:	0393	
			5E	DD	00022	PUSHL	SP	:		
		08	AE	9F	00024	PUSHAB	STV	:	0390	
			7E	7C	00027	CLRQ	-(SP)	:		
			56	DD	00029	PUSHL	R6	:		
		1C	AE	9F	0002B	PUSHAB	OUT_NAME	:		
			50	DD	0002E	PUSHL	IN_NAME	:	0391	
00000000G	00		07	FB	00030	CALLS	#7, LIB\$FIND_FILE	:		
	51		50	D0	00037	MOVL	RO, STATUS	:		
50	51	FFFF0007	8F	CB	0003A	BICL3	#-65529, STATUS, RO	:	0397	
00001128	8F		50	D1	00042	CMPL	RO, #4392	:		
			08	12	00049	BNEQ	1\$:		
	50	00000000G	8F	D0	0004B	MOVL	#CONVS_NOWILD, RO	:	0399	
			04	00052		RET		:		
	12		51	EB	00053	BLBS	STATUS, 2\$:	0403	
	50		66	D0	00056	MOVL	FINDFILEFAB, RO	:	0406	
	18	A0	00000000G	8F	D0	00059	MOVL	#CONVS_OPENIN, 24(RO)	:	
			50	DD	00061	PUSHL	RO	:	0407	
0000G	CF		01	FB	00063	CALLS	#1, CONV\$SRMS OPEN ERROR	:		
0000G	CF	08	AE	90	00068	MOVB	OUT_NAME, CONV\$AB_IN_FAB+52	:	0410	
0000G	CF	0C	AE	D0	0006E	MOVL	OUT_NAME+4, CONV\$AB_IN_FAB+44	:	0411	
		0000G	CF	B4	00074	CLRQ	CONVSAB_IN_FAB+2	:	0415	
	50		66	D0	00078	MOVL	FINDFILEFAB, RO	:	0416	
0000G	CF	40	A0	D0	0007B	MOVL	64(RO), CONV\$AB_IN_FAB+64	:		
	50	28	A0	D0	00081	MOVL	40(RO), FINDFILENAM	:	0418	
0000G	CF		14	A0	28	00085	MOVC3	#28, 20(FINDFILENAM), CONV\$AB_IN_NAM+20	:	0423
			50	01	D0	0008C	MOVL	#1, RO	:	0425
			04	0008F		RET		:	0427	

: Routine Size: 144 bytes, Routine Base: _CDNV\$CODE + 006B

: 433 0428 1

```

: 435 0429 1 %SBTTL 'OPEN_IN'
436 0430 1 GLOBAL ROUTINE CONV$$OPEN_IN =
437 0431 1 **
438 0432 1
439 0433 1 Functional Description:
440 0434 1
441 0435 1     Actually does the open of the input file, allocates and fills
442 0436 1     in key and area xabs if necessary, also connects record stream
443 0437 1
444 0438 1 Calling Sequence:
445 0439 1
446 0440 1     CONV$$OPEN_IN()
447 0441 1
448 0442 1 Input Parameters:
449 0443 1     none
450 0444 1
451 0445 1 Implicit Inputs:
452 0446 1
453 0447 1     CONV$AB_IN_FAB - Input fab
454 0448 1
455 0449 1 Output Parameters:
456 0450 1     none
457 0451 1
458 0452 1 Implicit Outputs:
459 0453 1     none
460 0454 1
461 0455 1 Routine Value:
462 0456 1
463 0457 1     CONV$_SUCCESS or CONV$_NOKEY
464 0458 1
465 0459 1 Routines Called:
466 0460 1
467 0461 1     $OPEN
468 0462 1     CONV$$RMS_OPEN_ERROR - By RMS as an AST
469 0463 1     CONV$$GET_VM
470 0464 1     $DISPLAY
471 0465 1     $CONNECT
472 0466 1
473 0467 1 Side Effects:
474 0468 1
475 0469 1     Opens and connects input file
476 0470 1
477 0471 1 --
478 0472 1
479 0473 2 BEGIN
480 0474 2
481 0475 2 LOCAL
482 0476 2     STATUS : LONG;
483 0477 2
484 0478 2     ! Set the FAB from the Option Switches
485 0479 2     !
486 0480 2     ! Read Check
487 0481 2     !
488 0482 2     CONV$AB_IN_FAB [ FAB$V_RCK ] = .CONV$GL_READ_C;
489 0483 2
490 0484 2     ! Input file sharing
491 0485 2

```



```

: 492      0486      2      IF .CONVSGL_SHARE
: 493      0487      2      THEN
: 494      0488      2      BEGIN
: 495      0489      2
: 496      0490      2      ! Set up the file sharing bits
: 497      0491      2      !
: 498      0492      2      CONVSAB_IN_FAB [ FAB$B_SHR ] = FAB$M_PUT OR FAB$M_GET OR FAB$M_DEL OR
: 499      0493      2      FAB$M_UPD OR FAB$M_UPI;
: 500      0494      2
: 501      0495      2      ! Do not wait for any record locks
: 502      0496      2      !
: 503      0497      2      CONVSAB_IN_RAB [ RAB$V_RRL ] = _SET
: 504      0498      2
: 505      0499      2      END;
: 506      0500      2
: 507      0501      2      ! If we have to access the file by key (other than primary) or we have
: 508      0502      2      to sort the file (which means we use RFA access)
: 509      0503      2      then clear the sqo bit
: 510      0504      2      !
: 511      0505      2      IF ( .CONVSGL_KEY NEQU 0 ) OR .CONVSGL_SORT
: 512      0506      2      THEN
: 513      0507      2      CONVSAB_IN_FAB [ FAB$V_SQO ] = _CLEAR;
: 514      0508      2
: 515      0509      2      ! Open the file
: 516      0510      2      !
: 517      0511      2      $OPEN ( FAB=CONVSAB_IN_FAB,ERR=CONV$$KMS_OPEN_ERROR );
: 518      0512      2
: 519      0513      2      ! Say that the file is open
: 520      0514      2      !
: 521      0515      2      CONVSAB_FLAGS [ CONVS$V_IN ] = _SET;
: 522      0516      2
: 523      0517      2      ! If this is an index file and we are creating the output file not by
: 524      0518      2      FDL definition then get the area ad key xabs
: 525      0519      2      !
: 526      0520      2      IF ( .CONVSAB_IN_FAB [ FAB$B_ORG ] EQLU FAB$C_IDX ) AND
: 527      0521      2      .CONVSGL_CREATE AND ( NOT .CONVSGL_FDL )
: 528      0522      2      THEN
: 529      0523      2      BEGIN
: 530      0524      2
: 531      0525      2      LOCAL
: 532      0526      2      BYTES,
: 533      0527      2      VM_POINTER,
: 534      0528      2      CURRENTXAB : REF BLOCK [ ,BYTE ];
: 535      0529      2
: 536      0530      2      BIND
: 537      0531      2      NEWXAB = VM_POINTER : REF BLOCK [ ,BYTE ];
: 538      0532      2
: 539      0533      2      ! Find out how much memory we need (The extra 32 is for the key name buffer)
: 540      0534      2      !
: 541      0535      2      BYTES = .CONVSAB_IN_XABSUM [ XAB$B_NOK ] * ( XAB$C_KEYLEN + 32 );
: 542      0536      2      BYTES = ( .CONVSAB_IN_XABSUM [ XAB$B_NOA ] * XAB$C_ALLLEN ) + .BYTES;
: 543      0537      2
: 544      0538      2      ! Get the address space
: 545      0539      2      !
: 546      0540      2      VM_POINTER = CONV$$GET_VM ( .BYTES );
: 547      0541      2
: 548      0542      2      ! The protection xab will point to the new xabs

```

```

549 0543 3
550 0544 3
551 0545 3
552 0546 3
553 0547 3
554 0548 3
555 0549 3
556 0550 3
557 0551 4
558 0552 4
559 0553 4
560 0554 4
561 0555 4
562 0556 4
563 0557 4
564 0558 4
565 0559 3
566 0560 3
567 0561 3
568 0562 3
569 0563 3
570 0564 3
571 0565 4
572 0566 4
573 0567 4
574 0568 4
575 0569 4
576 0570 4
577 0571 4
578 0572 3
579 0573 3
580 0574 3
581 0575 3
582 0576 3
583 0577 3
584 0578 3
585 0579 3
586 0580 4
587 0581 4
588 0582 2
589 0583 2
590 0584 2
591 0585 2
592 0586 2
593 0587 2
594 0588 2
595 0589 2
596 0590 2
597 0591 2
598 0592 2
599 0593 2
600 0594 2
601 0595 2
602 0596 2
603 0597 2
604 0598 2
605 0599 2

```

```

!
CURRENTXAB = CONV$AB_IN_XABSUM;
! Chain the xabs together and set up the fields
! Keys first
INCR I FROM 0 TO .CONV$AB_IN_XABSUM [ XAB$B_NOK ] - 1 BY 1
DO
  BEGIN
    CURRENTXAB [ XAB$B_NXT ] = .NEWXAB;
    CURRENTXAB = .NEWXAB;
    CURRENTXAB [ XAB$B_COD ] = XAB$C_KEY;
    CURRENTXAB [ XAB$B_BLN ] = XAB$C_KEYLEN;
    CURRENTXAB [ XAB$B_REF ] = .I;
    CURRENTXAB [ XAB$B_KNM ] = .CURRENTXAB + XAB$C_KEYLEN;
    NEWXAB = .NEWXAB + XAB$C_KEYLEN + 32
  END;
! Then areas
INCR I FROM 0 TO .CONV$AB_IN_XABSUM [ XAB$B_NOA ] - 1 BY 1
DO
  BEGIN
    CURRENTXAB [ XAB$B_NXT ] = .NEWXAB;
    CURRENTXAB = .NEWXAB;
    CURRENTXAB [ XAB$B_COD ] = XAB$C_ALL;
    CURRENTXAB [ XAB$B_BLN ] = XAB$C_ALLLEN;
    CURRENTXAB [ XAB$B_AID ] = .I;
    NEWXAB = .NEWXAB + XAB$C_ALLLEN
  END;
! The last xab points to 0
CURRENTXAB [ XAB$B_NXT ] = 0;
! Do a display to fill it all in
$DISPLAY ( FAB=CONV$AB_IN_FAB )
END;
! If this is an indexed file then set the key of ref. to input on
IF .CONV$AB_IN_FAB [ FAB$B_ORG ] EQL FAB$C_IDX
THEN
  ! If the key of ref. is out of range then signal an error and return
  ! normal. (so we can continue)
  IF .CONV$GL_KEY GEQ .CONV$AB_IN_XABSUM [ XAB$B_NOK ]
  THEN
    RETURN CONV$_NOKEY
  ELSE
    CONV$AB_IN_RAB [ RAB$B_KRF ] = .CONV$GL_KEY;
! Must Special Case for a UDF (Undefined) Input File
!

```

```

606 0600 2 IF .CONVSAB_IN_FAB [ FAB$B_RFM ] EQL FAB$C_UDF
607 0601 THEN
608 0602 BEGIN
609 0603
610 0604 : Get ready to input the file with Block IO
611 0605
612 0606 CONVSAB_IN_RAB [ RAB$L_BKT ] = 0;
613 0607 CONVSAB_IN_RAB [ RAB$V_BIO ] = _SET
614 0608 END
615 0609 ELSE
616 0610
617 0611 : Else do normal record IO
618 0612
619 0613 CONVSAB_IN_RAB [ RAB$V_BIO ] = _CLEAR;
620 0614
621 0615 : In normal operation IN_RAB points to IN_FAB but may be changed
622 0616 when doing sorts
623 0617
624 0618 CONVSAB_IN_RAB [ RAB$L_FAB ] = CONVSAB_IN_FAB;
625 0619
626 0620 : Now that every thing is ready connect a stream
627 0621
628 0622 $CONNECT ( RAB=CONVSAB_IN_RAB,ERR=CONV$$RMS_OPEN_ERROR );
629 0623
630 0624 : Any errors from now on are read errors
631 0625
632 0626 CONVSAB_IN_RAB [ RAB$L_CTX ] = CONV$_READERR;
633 0627
634 0628 RETURN CONV$_SUCCESS
635 0629
636 0630 END;

```

```

.EXTRN SYSSOPEN, SYSSDISPLAY
.EXTRN SYSSCONNECT

OFFC 00000

.ENTRY CONV$$OPEN_IN, Save R2,R3,R4,R5,R6,R7,R8,- : 0430
R9,R10,R11
MOVAB CONVSGL_KEY, R7
MOVAB CONVSAB_IN_XABSUM+9, R6
MOVAB CONVSAB_IN_RAB+4, R5
MOVAB CONVSAB_IN_FAB, R4
INSV CONVSGL_READ C, #7, #1, CONVSAB_IN_FAB+6 : 0482
BLBC CONVSGL_SHARE, 1$ : 0486
MOVB #79, CONVSAB_IN_FAB+23 : 0493
BISB2 #8, CONVSAB_IN_RAB+4 : 0497
TSTL CONVSGL_KEY : 0505
BNEQ 2$
BLBC CONVSGL_SORT, 3$
BICB2 #64, CONVSAB_IN_FAB+4 : 0507
PUSHAB CONV$$RMS_OPEN_ERROR : 0511
PUSHL R4
CALLS #2, SYSSOPEN
BISB2 #1, CONVSAB_FLAGS+2 : 0515
CMPB CONVSAB_IN_FAB+29, #32 : 0520
BNEQ 8$

```

06 A4

01

17

```

57 0000G CF 9E 00002
56 0000G CF 9E 00007
55 0000G CF 9E 0000C
54 0000G CF 9E 00011
07 0000G CF F0 00016
08 0000G CF E9 0001E
A4 4F 8F 90 00023
65 08 88 00028
67 D5 0002B 1$:
05 12 0002D
04 05 0000G CF E9 0002F
A4 40 8F 8A 00034 2$:
0000G CF 9F 00039 3$:
54 DD 0003D
00000000G 00 02 FB 0003F
0000G CF 01 88 00046
20 1D A4 91 0004B
7B 12 0004F

```

	76	0000G	CF	E9	00051	BLBC	CONV\$GL_CREATE, 8\$: 0521	
	71	0000G	CF	E8	00056	BLBS	CONV\$GL_FDL, 8\$: 0521	
	51		66	9A	0005B	MOVZBL	CONV\$AB_IN_XABSUM+9, BYTES	: 0535	
	51	0000006C	8F	C4	0005E	MULL2	#108, BYTES	: 0535	
	50	FF	A6	9A	00065	MOVZBL	CONV\$AB_IN_XABSUM+8, R0	: 0536	
	50		20	C4	00069	MULL2	#32, R0	: 0536	
	51		50	C0	0006C	ADDL2	R0, BYTES	: 0536	
			51	DD	0006F	PUSHL	BYTES	: 0540	
			0000G	30	00071	BSBW	CONV\$\$GET_VM	: 0540	
	5E		04	C0	00074	ADDL2	#4, SP	: 0540	
	51	F7	A6	9E	00077	MOVAB	CONV\$AB_IN_XABSUM, CURRENTXAB	: 0544	
	53		66	9A	0007B	MOVZBL	CONV\$AB_IN_XABSUM+9, R3	: 0549	
	52		01	CE	0007E	MNEGL	#1, I	: 0549	
			19	11	00081	BRB	5\$: 0549	
	04	A1	50	D0	00083	4\$: MOVL	NEWXAB, 4(CURRENTXAB)	: 0552	
	51		80	7E	00087	4\$: MOVAQ	(NEWXAB)+, CURRENTXAB	: 0553	
	61	4C15	8F	B0	0008A	4\$: MOVW	#19477, (CURRENTXAB)	: 0554	
	17	A1	52	90	0008F	4\$: MOVB	I, 23(CURRENTXAB)	: 0556	
	38	A1	4C	9E	00093	4\$: MOVAB	76(R1), 56(CURRENTXAB)	: 0557	
	50	64	A0	9E	00098	4\$: MOVAB	100(R0), NEWXAB	: 0558	
E3	52		53	F2	0009C	5\$: AOBLS	R3, I, 4\$: 0558	
	53	FF	A6	9A	000A0	5\$: MOVZBL	CONV\$AB_IN_XABSUM+8, R3	: 0563	
	52		01	CE	000A4	MNEGL	#1, I	: 0563	
			13	11	000A7	BRB	7\$: 0563	
	04	A1	50	D0	000A9	6\$: MOVL	NEWXAB, 4(CURRENTXAB)	: 0566	
	51		80	7E	000AD	6\$: MOVAQ	(NEWXAB)+, CURRENTXAB	: 0567	
	61	2014	8F	B0	000B0	6\$: MOVW	#8212, (CURRENTXAB)	: 0568	
	17	A1	52	90	000B5	6\$: MOVB	I, 23(CURRENTXAB)	: 0570	
	50		18	C0	000B9	6\$: ADDL2	#24, NEWXAB	: 0571	
E9	52		53	F2	000BC	7\$: AOBLS	R3, I, 6\$: 0571	
		04	A1	D4	000C0	7\$: CLRL	4(CURRENTXAB)	: 0576	
			54	DD	000C3	7\$: PUSHL	R4	: 0580	
	00000000G	00	01	FB	000C5	7\$: CALLS	#1, SYSSDISPLAY	: 0580	
	20	1D	A4	91	000CC	8\$: CMPB	CONV\$AB_IN_FAB+29, #32	: 0586	
			13	12	000D0	BNEQ	10\$: 0586	
67	66	08	00	ED	000D2	CMPZV	#0, #8, CONV\$AB_IN_XABSUM+9, CONV\$GL_KEY	: 0592	
			08	14	000D7	BGTR	9\$: 0592	
		50	00000000G	8F	D0	000D9	MOVL	#CONV\$_NOKEY, R0	: 0594
				04	000E0	RET		: 0594	
		31	A5	67	90	000E1	9\$: MOVB	CONV\$GL_KEY, CONV\$AB_IN_RAB+53	: 0596
			1F	A4	95	000E5	10\$: TSTB	CONV\$AB_IN_FAB+31	: 0600
				09	12	000E8	BNEQ	11\$: 0600
			34	A5	D4	000EA	CLRL	CONV\$AB_IN_RAB+56	: 0606
		01	A5	08	88	000ED	BISB2	#8, CONV\$AB_IN_RAB+5	: 0607
			04	11	000F1	BRB	12\$: 0607	
		01	A5	08	8A	000F3	11\$: BICB2	#8, CONV\$AB_IN_RAB+5	: 0613
		38	A5	64	9E	000F7	12\$: MOVAB	CONV\$AB_IN_FAB, CONV\$AB_IN_RAB+60	: 0618
			0000G	CF	9F	000FB	PUSHAB	CONV\$\$RMS_OPEN_ERROR	: 0622
			FC	A5	9F	000FF	PUSHAB	CONV\$AB_IN_RAB	: 0622
	00000000G	00	02	FB	00102	CALLS	#2, SYSSCONNECT	: 0622	
	14	A5	00000000G	8F	D0	00109	MOVL	#CONV\$_READERR, CONV\$AB_IN_RAB+24	: 0626
		50		01	D0	00111	MOVL	#1, R0	: 0628
				04	00114	RET		: 0630	

; Routine Size: 277 bytes, Routine Base: _CONV\$CODE + 00FB

```

638 0631 1 %SBTTL 'OPEN_OUTPUT'
639 0632 1 GLOBAL ROUTINE CONV$$OPEN_OUTPUT =
640 0633 1 ++
641 0634 1
642 0635 1 Functional Description:
643 0636 1
644 0637 1     Creates ( or opens ) the output file, connects a record stream and if
645 0638 1     it is an indexed file allocates and fills in the prologue key and
646 0639 1     area descriptor blocks for sort and/or fast load
647 0640 1
648 0641 1 Calling Sequence:
649 0642 1
650 0643 1     CONV$$OPEN_OUTPUT
651 0644 1
652 0645 1 Input Parameters:
653 0646 1     none
654 0647 1
655 0648 1 Implicit Inputs:
656 0649 1
657 0650 1     CONV$AB_OUT_FAB - Output fab
658 0651 1     CONV$AB_IN_FAB  - Input fab
659 0652 1     Option flags
660 0653 1
661 0654 1 Output Parameters:
662 0655 1     none
663 0656 1
664 0657 1 Implicit Outputs:
665 0658 1
666 0659 1     CONV$AB_FLAGS [ CONV$V_OUT ] - Set on success
667 0660 1
668 0661 1 Routine Value:
669 0662 1
670 0663 1     CONV$_SUCCESS or error from CONV$$OPEN_IN
671 0664 1
672 0665 1 Routines called:
673 0666 1
674 0667 1     $PARSE
675 0668 1     CONV$$RMS_OPEN_ERROR - By RMS as an AST
676 0669 1     $CREATE
677 0670 1     $DISPLAY
678 0671 1     $OPEN
679 0672 1     CONV$$READ_PROLOGUE
680 0673 1     $CONNECT
681 0674 1     CONV$$OPEN_IN
682 0675 1
683 0676 1 Side Effects:
684 0677 1     none
685 0678 1
686 0679 1 --
687 0680 1
688 0681 2 BEGIN
689 0682 2
690 0683 2 LOCAL
691 0684 2     PRESENT : LONG,
692 0685 2     OUT_DEV : BLOCK [ 1, LONG ],
693 0686 2     STATUS  : LONG;
694 0687 2

```

```

695 0688 2 ! Any rms errors on the output fab are OPENOUT errors
696 0689 2
697 0690 2 CONVSAB_OUT_FAB [ FABSL_CTX ] = CONVS_OPENOUT;
698 0691 2
699 0692 2 ! Use the file name in the call argument (not one from FDL)
700 0693 2
701 0694 2 CONVSAB_OUT_FAB [ FAB$B_FNS ] = .CONVSAB_OUT_FILE_NAM [ DSC$W_LENGTH ];
702 0695 2 CONVSAB_OUT_FAB [ FAB$B_FNA ] = .CONVSAB_OUT_FILE_NAM [ DSC$A_POINTER ];
703 0696 2
704 0697 2 ! Setup the input file name for output file related file parsing
705 0698 2
706 0699 2 CONVSAB_IN_NAM [ NAM$B_RSL ] = .CONVSAB_IN_FAB [ FAB$B_FNS ];
707 0700 2 CONVSAB_IN_NAM [ NAM$B_RSA ] = .CONVSAB_IN_FAB [ FAB$B_FNA ];
708 0701 2
709 0702 2 ! Parse the name
710 0703 2
711 0704 2 $PARSE ( FAB=CONVSAB_OUT_FAB,ERR=CONVS$RMS_OPEN_ERROR );
712 0705 2
713 0706 2 ! Check the device that the output file is on
714 0707 2
715 0708 2 OUT_DEV = .CONVSAB_OUT_FAB [ FAB$B_DEV ];
716 0709 2
717 0710 2 ! We cannot fast load to a network device
718 0711 2
719 0712 2 IF .OUT_DEV [ DEV$V_NET ]
720 0713 2 THEN
721 0714 2     CONVSGL_FAST = _CLEAR;
722 0715 2
723 0716 2 ! Set the FAB from the Option Switches
724 0717 2
725 0718 2 ! Write Check
726 0719 2
727 0720 2 CONVSAB_OUT_FAB [ FAB$V_WCK ] = .CONVSGL_WRITE_C;
728 0721 2
729 0722 2 ! Merge
730 0723 2
731 0724 2 CONVSAB_OUT_FAB [ FAB$V_SQO ] = ( NOT .CONVSGL_MERGE )
732 0725 2
733 0726 2 ! If the CREATE Option was specified then Create the output file
734 0727 2 ! else just open it
735 0728 2
736 0729 2 IF .CONVSGL_CREATE
737 0730 2 THEN
738 0731 2     BEGIN
739 0732 2
740 0733 2     LOCAL COPY_FAB : REF BLOCK [ ,BYTE ];
741 0734 2
742 0735 2     ! Determine where to copy fab from
743 0736 2
744 0737 2     IF .CONVSGL_FDL
745 0738 2     THEN
746 0739 2         BEGIN
747 0740 2
748 0741 2         ! If fdl was done copy the stuff from the fab produced by
749 0742 2         ! fdl$parse
750 0743 2
751 0744 2         COPY_FAB = .CONVSAB_FDL_FAB;

```

```

752 0745 4
753 0746 4      ! Connect the fdl xabs
754 0747 4
755 0748 4      CONV$AB_OUT_XABSUM [ XAB$L_NXT ] = .COPY_FAB [ FAB$L_XAB ]
756 0749 4
757 0750 4      END
758 0751 3      ELSE
759 0752 4      BEGIN
760 0753 4
761 0754 4      ! If this is not a create by FDL definition then get the stuff
762 0755 4      ! from the input file
763 0756 4
764 0757 4      COPY_FAB = CONV$AB_IN_FAB;
765 0758 4
766 0759 4      ! Connect the input files summary xab NXT which will connect
767 0760 4      ! any other xabs that the input file may have had ie. area and
768 0761 4      ! key xabs
769 0762 4
770 0763 4      CONV$AB_OUT_XABSUM [ XAB$L_NXT ] = .CONV$AB_IN_XABSUM [ XAB$L_NXT ]
771 0764 4
772 0765 3      END;
773 0766 3
774 0767 3      ! Copy the important fab fields
775 0768 3
776 0769 3      CONV$AB_OUT_FAB [ FAB$L_ALQ ] = .COPY_FAB [ FAB$L_ALQ ]; ! Allocation
777 0770 3      CONV$AB_OUT_FAB [ FAB$W_DEQ ] = .COPY_FAB [ FAB$W_DEQ ]; ! Extension
778 0771 3      CONV$AB_OUT_FAB [ FAB$B_RTV ] = .COPY_FAB [ FAB$B_RTV ]; ! Reteval windoow
779 0772 3      CONV$AB_OUT_FAB [ FAB$B_ORG ] = .COPY_FAB [ FAB$B_ORG ]; ! Organization
780 0773 3      CONV$AB_OUT_FAB [ FAB$B_RAT ] = .COPY_FAB [ FAB$B_RAT ]; ! Record attributes
781 0774 3      CONV$AB_OUT_FAB [ FAB$B_RFM ] = .COPY_FAB [ FAB$B_RFM ]; ! Record format
782 0775 3      CONV$AB_OUT_FAB [ FAB$W_MRS ] = .COPY_FAB [ FAB$W_MRS ]; ! Max record size
783 0776 3      CONV$AB_OUT_FAB [ FAB$L_MRN ] = .COPY_FAB [ FAB$L_MRN ]; ! Max records
784 0777 3      CONV$AB_OUT_FAB [ FAB$W_BLS ] = .COPY_FAB [ FAB$W_BLS ]; ! Block size
785 0778 3      CONV$AB_OUT_FAB [ FAB$B_BKS ] = .COPY_FAB [ FAB$B_BKS ]; ! Bucket size
786 0779 3      CONV$AB_OUT_FAB [ FAB$B_FSZ ] = .COPY_FAB [ FAB$B_FSZ ]; ! Fixed size
787 0780 3      CONV$AB_OUT_FAB [ FAB$W_GBC ] = .COPY_FAB [ FAB$W_GBC ]; ! Global Buffers
788 0781 3
789 0782 3      CONV$AB_OUT_FAB [ FAB$L_FOP ] = .CONV$AB_OUT_FAB [ FAB$L_FOP ] OR
790 0783 3      ! .COPY_FAB [ FAB$L_FOP ]; ! File options
791 0784 3
792 0785 3      ! If the PROLOGUE option was specified and the file is indexed
793 0786 3      ! then stuff the first key xab with the user value
794 0787 3
795 0788 3      IF ( .CONV$AB_OUT_FAB [ FAB$B_ORG ] EQLU FAB$C_IDX ) AND
796 0789 3      ! .CONV$AB_FLAGS [ CONV$V_PROLOG ]
797 0790 3      THEN
798 0791 4      BEGIN
799 0792 4
800 0793 4      LOCAL XAB : REF BLOCK [ ,BYTE ];
801 0794 4
802 0795 4      ! Find the first key xab
803 0796 4
804 0797 4      XAB = .CONV$AB_OUT_FAB [ FAB$L_XAB ];
805 0798 4
806 0799 4      ! The xabs have to be in order and there must be a key 0 so
807 0800 4      ! the first one we find is the one we want
808 0801 4

```

```

809      0802      4      WHILE .XAB [ XAB$B_COD ] NEQU XAB$C_KEY
810      0803      4      DO
811      0804      4
812      0805      4      ! If there are no more xabs then we really have a problem
813      0806      4      ! so forget it
814      0807      4
815      0808      4      IF .XAB [ XAB$L_NXT ] EQLU 0
816      0809      4      THEN
817      0810      4      RETURN CONV$BADLOGIC
818      0811      4      ELSE
819      0812      4      XAB = .XAB [ XAB$L_NXT ];
820      0813      4
821      0814      4      ! Stuff the value
822      0815      4      !
823      0816      4      XAB [ XAB$B_PROLOG ] = .CONV$GL_PROLOG
824      0817      4
825      0818      4      END;
826      0819      4
827      0820      4      ! Create it
828      0821      4      !
829      0822      4      ! If the record format was changed on a non VMS system
830      0823      4      ! signal a warning (only to DLL)
831      0824      4
832      0825      4      $CREATE ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
833      0826      4
834      0827      4      IF ( $CREATE ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR ) EQLU
835      0828      4      RMS$ (RE_STM ) AND
836      0829      4      .CONV$AB_FLAGS [ -CONV$V_DCL ]
837      0830      4      THEN
838      0831      4      SIGNAL ( CONV$_CREATEDSTM );
839      0832      4
840      0833      4      ! Since a create does not fill in the summary xab do a display
841      0834      4      !
842      0835      4      $DISPLAY( FAB=CONV$AB_OUT_FAB )
843      0836      4
844      0837      4      END
845      0838      4      ELSE
846      0839      4      $OPEN ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
847      0840      4
848      0841      4      ! If we got here then we have opened a file.
849      0842      4      !
850      0843      4      CONV$AB_FLAGS [ CONV$V_OUT ] = _SET;
851      0844      4
852      0845      4      ! Set some bits depending on the type of output file
853      0846      4      !
854      0847      4      ! Can only append to a sequential file
855      0848      4      !
856      0849      4      IF .CONV$AB_OUT_FAB [ FAB$B_ORG ] EQLU FAB$C_SEQ
857      0850      4      THEN
858      0851      4      CONV$AB_OUT_RAB [ RAB$V_EOF ] = .CONV$GL_APPEND
859      0852      4      ELSE
860      0853      4
861      0854      4      ! If append was on without a seq. output file then error
862      0855      4      !
863      0856      4      IF .CONV$GL_APPEND THEN RETURN CONV$_NOTSEQ;
864      0857      4
865      0858      4      ! Is't not very exciting if it's not an index file

```



```

866 0859 2 !
867 0860 2 IF .CONV$AB_OUT_FAB [ FAB$B_ORG ] NEQU FAB$C_IDX
868 0861 2 THEN
869 0862 2 BEGIN
870 0863 2 CONV$GL_MERGE = _CLEAR;
871 0864 2 CONV$GL_SORT = _CLEAR;
872 0865 2 CONV$GL_FAST = _CLEAR;
873 0866 2 END
874 0867 2 ELSE
875 0868 2
876 0869 2 ! Set the fill option if it is indexed
877 0870 2
878 0871 2 CONV$AB_OUT_RAB [ RAB$V_LOA ] = NOT .CONV$GL_FILL;
879 0872 2
880 0873 2 ! If we are sorting or fastloading
881 0874 2 then allocate space for KEY and AREA XAB's and fill them in by reading
882 0875 2 the prologue blocks in the file
883 0876 2
884 0877 2 IF ( .CONV$GL_FAST OR .CONV$GL_SORT )
885 0878 2 THEN
886 0879 2 BEGIN
887 0880 2
888 0881 2 ! Connect the file for Block IO for reading the
889 0882 2 prologue.
890 0883 2
891 0884 2 CONV$AB_OUT_RAB [ RAB$V_BIO ] = _SET;
892 0885 2
893 0886 2 $CONNECT ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_OPEN_ERROR );
894 0887 2
895 0888 2 ! Read the prologue
896 0889 2
897 0890 2 CONV$$READ_PROLOGUE();
898 0891 2
899 0892 2 ! If this is not a fast load then we need to bounce the file so we can
900 0893 2 do record IO again. (This sure doesn't look good!)
901 0894 2
902 0895 2 IF NOT .CONV$GL_FAST
903 0896 2 THEN
904 0897 2 BEGIN
905 0898 2
906 0899 2 ! Disconnect and Close (Dont check the disconnect)
907 0900 2
908 0901 2 $DISCONNECT ( RAB=CONV$AB_OUT_RAB );
909 0902 2 $CLOSE( FAB=CONV$AB_OUT_FAB );
910 0903 2
911 0904 2 ! Clear the Block IO flag
912 0905 2
913 0906 2 CONV$AB_OUT_RAB [ RAB$V_BIO ] = _CLEAR;
914 0907 2
915 0908 2 ! Reopen and Reconnect (Dont need to reconnect the PLG RAB)
916 0909 2
917 0910 2 $OPEN ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_OPEN_ERROR );
918 0911 2
919 0912 2 $CONNECT ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_OPEN_ERROR )
920 0913 2
921 0914 2 END
922 0915 2 END

```

```

: 923 0916 2 ELSE
: 924 0917 BEGIN
: 925 0918
: 926 0919 : If we are merging into an indexed file
: 927 0920 : then set the access to KEY
: 928 0921
: 929 0922 IF .CONV$GL_MERGE
: 930 0923 THEN
: 931 0924 CONV$AB_OUT_RAB [ RAB$B_RAC ] = RAB$C_KEY;
: 932 0925
: 933 0926 : If we are not sorting or fastloading
: 934 0927 : then connect the stream normally
: 935 0928
: 936 0929 $CONNECT ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_OPEN_ERROR );
: 937 0930
: 938 0931 : If the output file was not opened by now we can open it here
: 939 0932
: 940 0933 IF NOT .CONV$AB_FLAGS [ CONV$V_IN ]
: 941 0934 THEN
: 942 0935 RET_ON_ERROR( CONV$$OPEN_IN() )
: 943 0936
: 944 0937 END;
: 945 0938
: 946 0939 ! If PAD switch is on and the file is not fixed format
: 947 0940
: 948 0941 IF .CONV$GL_PAD AND ( .CONV$AB_OUT_FAB [ FAB$B_RFM ] NEQU FAB$C_FIX )
: 949 0942 THEN
: 950 0943 BEGIN
: 951 0944 CONV$GL_PAD = CLEAR;
: 952 0945 SIGNAL( CONV$_PAD )
: 953 0946 END;
: 954 0947
: 955 0948 ! Any errors on the output rab should be write errors (exceptions are in
: 956 0949 ! the fast load code
: 957 0950
: 958 0951 CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
: 959 0952
: 960 0953 ! Return normally
: 961 0954
: 962 0955 RETURN CONV$_SUCCESS
: 963 0956
: 964 0957 1 END;

```

```

.EXTRN SYSSPARSE, SYSSCREATE
.EXTRN SYSSDISCONNECT, SYSSCLOSE

```

OFFC 00000

```

59 0000G CF 9E 00002
58 0000G CF 9E 00007
57 00000000G 00 9E 0000C
56 0000G CF 9E 00013
55 00000000G 00 9E 00018
54 0000G CF 9E 0001F
53 0000G CF 9E 00024

```

```

.ENTRY CONV$$OPEN_OUTPUT, Save R2,R3,R4,R5,R6,R7,- ; 0632
R8,R9,R10,R11
MOVAB CONV$AB_FLAGS+2, R9
MOVAB CONV$GL_MERGE, R8
MOVAB SYSSOPEN, R7
MOVAB CONV$GL_FAST, R6
MOVAB SYSSCONNECT, R5
MOVAB CONV$$RMS_OPEN_ERROR, R4
MOVAB CONV$AB_OUT_RAB+4, R3

```

		52	0000G	CF	9E	00029	MOVAB	CONVSAB_OUT_FAB, R2	...			
	18	A2	00000000G	8F	D0	0002E	MOVL	#CONVS_OPENOUT, CONVSAB_OUT_FAB+24	...	0690		
		50	0000G	CF	D0	00036	MOVL	CONVSAB_OUT_FILE_NAM, R0	...	0694		
	34	A2		60	90	0003B	MOVW	(R0), CONVSAB_OUT_FAB+52	...			
	2C	A2	04	A0	D0	0003F	MOVL	4(R0), CONVSAB_OUT_FAB+44	...	0695		
	0000G	CF	0000G	CF	90	00044	MOVW	CONVSAB_IN_FAB+52, CONVSAB_IN_NAM+3	...	0699		
	0000G	CF	0000G	CF	D0	0004B	MOVL	CONVSAB_IN_FAB+44, CONVSAB_IN_NAM+4	...	0700		
		00000000G	00	14	BB	00052	PUSHR	#*M<R2,R4>	...	0704		
			50	02	FB	00054	CALLS	#2, SYSSPARSE	...			
	02	50	40	A2	D0	0005B	MOVL	CONVSAB_OUT_FAB+64, OUT_DEV	...	0708		
				0D	E1	0005F	BBC	#13, OUT_DEV, 1\$...	0712		
				66	D4	00063	CLRL	CONVSGL_FAST	...	0714		
05	A2	01	01	0000G	CF	F0	00065	1\$: INSV	CONVSGL_WRITE_C, #1, #1, CONVSAB_OUT_FAB+5	...	0720	
			50	68	D2	0006D	MCOML	CONVSGL_MERGE, R0	...	0724		
04	A2	01	06	50	F0	00070	INSV	R0, #6, #1, CONVSAB_OUT_FAB+4	...			
			03	0000G	CF	E8	00076	BLBS	CONVSGL_CREATE, 2\$...	0729	
				00B1	31	0007B	BRW	9\$...			
			0D	0000G	CF	E9	0007E	2\$: BLBC	CONVSGL_FDL, 3\$...	0737	
			50	0000	CF	D0	00083	MOVL	CONVSAB_FDL_FAB, COPY_FAB	...	0744	
	0000G	CF	24	A0	D0	00088	MOVL	36(COPY_FAB), CONVSAB_OUT_XABSUM+4	...	0748		
				0C	11	0008E	BRB	4\$...			
			50	0000G	CF	9E	00090	3\$: MOVAB	CONVSAB_IN_FAB, COPY_FAB	...	0757	
	0000G	CF	0000G	CF	D0	00095	MOVL	CONVSAB_IN_XABSUM+4, CONVSAB_OUT_XABSUM+4	...	0763		
	10	A2	10	A0	D0	0009C	4\$: MOVL	16(COPY_FAB), CONVSAB_OUT_FAB+16	...	0769		
	14	A2	14	A0	B0	000A1	MOVW	20(COPY_FAB), CONVSAB_OUT_FAB+20	...	0770		
	1C	A2	1C	A0	D0	000A6	MOVL	28(COPY_FAB), CONVSAB_OUT_FAB+28	...	0771		
	36	A2	36	A0	B0	000AB	MOVW	54(COPY_FAB), CONVSAB_OUT_FAB+54	...	0775		
	38	A2	38	A0	7D	000B0	MOVW	56(COPY_FAB), CONVSAB_OUT_FAB+56	...	0776		
	48	A2	48	A0	B0	000B5	MOVW	72(COPY_FAB), CONVSAB_OUT_FAB+72	...	0780		
	04	A2	04	A0	C8	000BA	BISL2	4(COPY_FAB), CONVSAB_OUT_FAB+4	...	0783		
			20	1D	A2	91	000BF	CMPB	CONVSAB_OUT_FAB+29, #32	...	0788	
				26	12	000C3	BNEW	8\$...			
	22		69	06	E1	000C5	BBC	#6, CONVSAB_FLAGS+2, 8\$...	0789		
			50	24	A2	D0	000C9	MOVL	CONVSAB_OUT_FAB+36, XAB	...	0797	
			15	60	91	000CD	5\$: CMPB	(XAB), #21	...	0802		
				13	13	C00D0	BEQL	7\$...			
				04	A0	D5	000D2	TSTL	4(XAB)	...	0808	
				08	12	000D5	BNEQ	6\$...			
			50	00000000G	8F	D0	000D7	MOVL	#CONVS_BADLOGIC, R0	...	0810	
				04	000DE		RET		...			
			50	04	A0	D0	000DF	6\$: MOVL	4(XAB), XAB	...	0812	
				E8	11	000E3	BRB	5\$...	0808		
			48	A0	0000G	CF	90	000E5	7\$: MOVW	CONVSGL_PROLOG, 72(XAB)	...	0816
				14	BB	000EB	8\$: PUSHR	#*M<R2,R4>	...	0825		
	00000000G	00		02	FB	000ED	CALLS	#2, SYSSCREATE	...			
	00000000G	00		52	DD	000F4	PUSHL	R2	...	0835		
				01	FB	000F6	CALLS	#1, SYSSDISPLAY	...			
				05	11	000FD	BRB	10\$...			
				14	BB	000FF	9\$: PUSHR	#*M<R2,R4>	...	0839		
			67	02	FB	00101	CALLS	#2, SYSSOPEN	...			
			69	02	88	00104	10\$: BISB2	#2, CONVSAB_FLAGS+2	...	0843		
			50	1D	A2	9A	00107	MOVZBL	CONVSAB_OUT_FAB+29, R0	...	0849	
				0A	12	0010B	BNEQ	11\$...			
01	A3	01	00	0000G	CF	F0	0010D	INSV	CONVSGL_APPEND, #0, #1, CONVSAB_OUT_RAB+5	...	0851	
				0D	11	00115	BRB	12\$...			
			08	0000G	CF	E9	00117	11\$: BLBC	CONVSGL_APPEND, 12\$...	0856	
			50	00000000G	8F	D0	0011C	MOVL	#CONVS_NOTSEQ, R0	...		

				04 00123	RET		
			20	50 91 00124	CMPB	R0, #32	0860
				0A 13 00127	BEQL	13\$	
				68 D4 00129	CLRL	CONV\$GL_MERGE	0863
				0000G CF D4 0012B	CLRL	CONV\$GL_SORT	0864
				66 D4 0012F	CLRL	CONV\$GL_FAST	0865
				0B 11 00131	BRB	14\$	
			50	0000G CF D2 00133	MCOML	CONV\$GL_FILL, R0	0871
01	A3		05	50 F0 00138	INSV	R0, #5, #1, CONV\$AB_OUT_RAB+5	
			05	66 E8 0013E	BLBS	CONV\$GL_FAST, 15\$	0877
			38	0000G CF E9 00141	BLBC	CONV\$GL_SORT, 16\$	
			01	A3 08 88 00146	BISB2	#8, CONV\$AB_OUT_RAB+5	0884
				54 DD 0014A	PUSHL	R4	0886
				FC A3 9F 0014C	PUSHAB	CONV\$AB_OUT_RAB	
			65	02 FB 0014F	CALLS	#2, SYS\$CONNECT	
				0000G 30 00152	BSBW	CONV\$\$READ_PROLOGUE	0890
			40	66 E8 00155	BLBS	CONV\$GL_FAST, 18\$	0895
				FC A3 9F 00158	FJSHAB	CONV\$AB_OUT_RAB	0901
		00000000G	00	01 FB 0015B	CALL	#1, SYS\$DISCONNECT	
				52 DD 00162	PUSH	R2	0902
		00000000G	00	01 FB 00164	CALLS	#1, SYS\$CLOSE	
		01	A3	08 8A 0016B	BICB2	#8, CONV\$AB_OUT_RAB+5	0906
				14 BB 0016F	PUSHP	#*M<R2, R4>	0910
			67	02 FB 00171	CALLS	#2, SYS\$OPEN	
				54 DD 00174	PUSHL	R4	0912
				FC A3 9F 00176	PUSHAB	CONV\$AB_OUT_RAB	
			65	02 FB 00179	CALLS	#2, SYS\$CONNECT	
				1A 11 0017C	BRB	18\$	0895
			04	68 E9 0017E	BLBC	CONV\$GL_MERGE, 17\$	0922
			1A	A3 01 90 00181	MOVB	#1, CONV\$AB_OUT_RAB+30	0924
				54 DD 00185	PUSHL	R4	0929
				FC A3 9F 00187	PUSHAB	CONV\$AB_OUT_RAB	
			65	02 FB 0018A	CALLS	#2, SYS\$CONNECT	
			08	69 E8 0018D	BLBS	CONV\$AB_FLAGS+2, 18\$	0933
		FD56		CF 00 FB 00190	CALLS	#0, CONV\$\$OPEN_IN	0935
			27	50 E9 00195	BLBC	STATUS, 20\$	
			17	0000G CF E9 00198	BLBC	CONV\$GL_PAD, 19\$	0941
			01	1F A2 91 0019D	CMPB	CONV\$AB_OUT_FAB+31, #1	
				11 13 001A1	BEQL	19\$	
				0000G CF D4 001A3	CLRL	CONV\$GL_PAD	0944
		00000000G	00	8F DD 001A7	PUSHL	#CONV\$ PAD	0945
			14	01 FB 001AD	CALLS	#1, LIB\$SIGNAL	
		00000000G	A3	00000000G 8F D0 001B4	MOVL	#CONV\$_WRITEERR, CONV\$AB_OUT_RAB+24	0951
			50	01 D0 001BC	MOVL	#1, R0	0955
				04 001BF	RET		0957

: Routine Size: 448 bytes, Routine Base: _CONV\$CODE + 0210

```

: 966 0958 1 %SBTTL 'CREATE_BUFFER'
: 967 0959 1 GLOBAL ROUTINE 'CONV$$CREATE_BUFFER =
: 968 0960 1 ++
: 969 0961 1
: 970 0962 1 Functional Description:
: 971 0963 1
: 972 0964 1     Creates the main record buffer and sets the record sizes
: 973 0965 1
: 974 0966 1     The main record buffer:
: 975 0967 1
: 976 0968 1           -----//-----
: 977 0969 1           |                                     |
: 978 0970 1           |                                     |
: 979 0971 1           |                                     |
: 980 0972 1           |                                     |
: 981 0973 1           |                                     |
: 982 0974 1           |                                     |
: 983 0975 1           |                                     |
: 984 0976 1           |                                     |
: 985 0977 1           |                                     |
: 986 0978 1           |                                     |
: 987 0979 1           |                                     |
: 988 0980 1           |                                     |
: 989 0981 1           |                                     |
: 990 0982 1           |                                     |
: 991 0983 1           |                                     |
: 992 0984 1           |                                     |
: 993 0985 1           |                                     |
: 994 0986 1           |                                     |
: 995 0987 1           |                                     |
: 996 0988 1           |                                     |
: 997 0989 1           |                                     |
: 998 0990 1           |                                     |
: 999 0991 1           |                                     |
1000 0992 1           |                                     |
1001 0993 1           |                                     |
1002 0994 1           |                                     |
1003 0995 1           |                                     |
1004 0996 1           |                                     |
1005 0997 1           |                                     |
1006 0998 1           |                                     |
1007 0999 1           |                                     |
1008 1000 1           |                                     |
1009 1001 1           |                                     |
1010 1002 1           |                                     |
1011 1003 1           |                                     |
1012 1004 1           |                                     |
1013 1005 1           |                                     |
1014 1006 1           |                                     |
1015 1007 2           |                                     |
1016 1008 2           |                                     |
1017 1009 2           |                                     |
1018 1010 2           |                                     |
1019 1011 2           |                                     |
1020 1012 2           |                                     |
1021 1013 2           |                                     |
: 1022 1014 2           |                                     |

```

VFC_BUF_PTR REC_BUF_PTR

```

: 985 0977 1 Calling Sequence:
: 986 0978 1     CONV$$CREATE_BUFFER()
: 987 0979 1
: 988 0980 1 Input Parameters:
: 989 0981 1     none
: 990 0982 1
: 991 0983 1 Implicit Inputs:
: 992 0984 1     none
: 993 0985 1
: 994 0986 1 Output Parameters:
: 995 0987 1     none
: 996 0988 1
: 997 0989 1 Implicit Outputs:
: 998 0990 1     Record buffer pointers and size variables
: 999 0991 1
1000 0992 1
1001 0993 1 Routine Value:
1002 0994 1     $$$_NORMAL
1003 0995 1
1004 0996 1 Routines Called:
1005 0997 1     CONV$$GET_VM
1006 0998 1
1007 0999 1 Side Effects:
1008 1000 1     none
1009 1001 1
1010 1002 1
1011 1003 1
1012 1004 1
1013 1005 1
1014 1006 1
1015 1007 2 BEGIN
1016 1008 2
1017 1009 2 LITERAL
1018 1010 2     MAX_REC_CTRL = 14; ! Maximun number of control bytes for a data
1019 1011 2     ! record in index file (14 is for a fully
1020 1012 2     ! compressed prologue 3 record
1021 1013 2
: 1022 1014 2 LOCAL

```

```

1023      1015      2          IN_VFC,
1024      1016      2          IN_MRS,
1025      1017      2          OUT_VFC,
1026      1018      2          OUT_EXTRA;
1027      1019      2
1028      1020      2          ! Account for the VFC temporarily
1029      1021      2
1030      1022      2          IF .CONVSAB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_VFC
1031      1023      2          THEN
1032      1024      2              OUT_VFC = .CONVSAB_OUT_FAB [ FAB$B_FSZ ]
1033      1025      2          ELSE
1034      1026      2              OUT_VFC = 0;
1035      1027      2
1036      1028      2          ! If output MRS = 0 ( ie. VAR and VFC records ) then
1037      1029      2          ! check for Block Spanning with Sequential Files
1038      1030      2          ! and Bucket Crossing with Relative and Indexed
1039      1031      2
1040      1032      2          IF ( CONV$GW_OUT_MRS = .CONVSAB_OUT_FAB [ FAB$W_MRS ] ) EQL 0
1041      1033      2          THEN
1042      1034      2              BEGIN
1043      1035      2
1044      1036      2              LOCAL   OUT_DEV : BLOCK [ 1, LONG ];
1045      1037      2
1046      1038      2              ! Find out if this thing is going to tape, if so use block size
1047      1039      2              ! (Since records cannot spand blocks on tape)
1048      1040      2
1049      1041      2              OUT_DEV = .CONVSAB_OUT_FAB [ FAB$L_DEV ];
1050      1042      2
1051      1043      2              IF .OUT_DEV [ DEV$V_SQD ]
1052      1044      2              THEN
1053      1045      2                  CONV$GW_OUT_MRS = .CONVSAB_OUT_FAB [ FAB$W_BLS ] - .OUT_VFC - 2
1054      1046      2
1055      1047      2              ! Sequential and NO Block spanning
1056      1048      2
1057      1049      2              ELSE IF ( .CONVSAB_OUT_FAB [ FAB$B_ORG ] EQLU FAB$C_SEQ ) AND
1058      1050      2                  .CONVSAB_OUT_FAB [ FAB$V_BLK ]
1059      1051      2              THEN
1060      1052      2                  CONV$GW_OUT_MRS = BLOCK_SIZE - .OUT_VFC - 2
1061      1053      2
1062      1054      2              ! Relative
1063      1055      2
1064      1056      2              ELSE IF .CONVSAB_OUT_FAB [ FAB$B_ORG ] EQLU FAB$C_REL
1065      1057      2              THEN
1066      1058      2                  CONV$GW_OUT_MRS = ( .CONVSAB_OUT_FAB [ FAB$B_BKS ] * BLOCK_SIZE ) -
1067      1059      2                  .OUT_VFC - 3
1068      1060      2
1069      1061      2              ! Indexed
1070      1062      2
1071      1063      2              ELSE
1072      1064      2                  CONV$GW_OUT_MRS = ( .CONVSAB_OUT_FAB [ FAB$B_BKS ] * BLOCK_SIZE ) -
1073      1065      2                  .OUT_VFC - 7;
1074      1066      2
1075      1067      2              END;
1076      1068      2
1077      1069      2          ! If the Input File is UDF then the UDF_MRS is calculated from
1078      1070      2          ! the output file attributes
1079      1071      2

```

```

: 1080      1072      2      IF .CONVSAB_IN_FAB [ FAB$B_RFM ] EQLU FAB$C_UDF
: 1081      1073      2      THEN
: 1082      1074      2      BEGIN
: 1083      1075      2
: 1084      1076      2      IN_MRS = BLOCK_SIZE;
: 1085      1077      2
: 1086      1078      2      ! If fixed format then no problem use that value, if
: 1087      1079      2      ! not see if a 512 byte record will fit
: 1088      1080      2
: 1089      1081      2      IF .CONVSAB_OUT_FAB [ FAB$B_RFM ] EQL FAB$C_FIX
: 1090      1082      2      THEN
: 1091      1083      2      CONV$GW_UDF_MRS = .CONVSAB_OUT_FAB [ FAB$W_MRS ]
: 1092      1084      2      ELSE
: 1093      1085      2
: 1094      1086      2      ! If the udf record will not fit into the output file then error
: 1095      1087      2
: 1096      1088      2      IF .CONVS$GW_OUT_MRS LSS BLOCK_SIZE
: 1097      1089      2      THEN
: 1098      1090      2      RETURN CONV$UDF_BLK
: 1099      1091      2      ELSE
: 1100      1092      2      CONV$GW_UDF_MRS = BLOCK_SIZE
: 1101      1093      2      END
: 1102      1094      2      ELSE
: 1103      1095      2      BEGIN
: 1104      1096      2
: 1105      1097      2      ! Here for a normal input file
: 1106      1098      2      ! IN_MRS is the length of the maximum record size
: 1107      1099      2
: 1108      1100      2      ! Now see if the file is VFC
: 1109      1101      2
: 1110      1102      2      IF .CONVSAB_IN_FAB [ FAB$B_RFM ] EQL FAB$C_VFC
: 1111      1103      2      THEN
: 1112      1104      2      IN_VFC = .CONVSAB_IN_FAB [ FAB$B_FSZ ]
: 1113      1105      2      ELSE
: 1114      1106      2      IN_VFC = 0;
: 1115      1107      2
: 1116      1108      2      ! If max. record size is zero then we find out from Longest Record Length
: 1117      1109      2      ! on disk or Block Size for magtape
: 1118      1110      2
: 1119      1111      2      IF ( IN_MRS = .CONVSAB_IN_FAB [ FAB$W_MRS ] ) EQL 0
: 1120      1112      2      THEN
: 1121      1113      2      BEGIN
: 1122      1114      2      LOCAL IN_DEV : BLOCK [ 1, LONG ];
: 1123      1115      2
: 1124      1116      2      ! Find out if this thing is coming from tape if so use block size
: 1125      1117      2      ! (Since records cannot span blocks on tape)
: 1126      1118      2
: 1127      1119      2      IN_DEV = .CONVSAB_IN_FAB [ FAB$L_DEV ];
: 1128      1120      2
: 1129      1121      2      IF .IN_DEV [ DEV$V_SQD ]
: 1130      1122      2      THEN
: 1131      1123      2      IN_MRS = .CONVSAB_IN_FAB [ FAB$W_BLS ] - .IN_VFC
: 1132      1124      2
: 1133      1125      2
: 1134      1126      2      ! If SEQ use LRL otherwise check
: 1135      1127      2      ! bucket sizes
: 1136      1128      2

```

```

: 1137      1129      4      ELSE IF .CONVSAB_IN_FAB [ FAB$B_ORG ] EQL FAB$C_SEQ
: 1138      1130      4      THEN
: 1139      1131      4      IN_MRS = .CONVSAB_IN_XABFHC [ XAB$W_LRL ]
: 1140      1132      4
: 1141      1133      4      ! Relative
: 1142      1134      4      !
: 1143      1135      4      ELSE IF .CONVSAB_IN_FAB [ FAB$B_ORG ] EQL FAB$C_REL
: 1144      1136      4      THEN
: 1145      1137      4      IN_MRS = ( .CONVSAB_IN_FAB [ FAB$B_BKS ] * BLOCK_SIZE ) -
: 1146      1138      4      .IN_VFC - 3
: 1147      1139      4
: 1148      1140      4      ! Indexed
: 1149      1141      4      !
: 1150      1142      4      ELSE
: 1151      1143      4      IN_MRS = ( .CONVSAB_IN_FAB [ FAB$B_BKS ] * BLOCK_SIZE ) -
: 1152      1144      4      .IN_VFC - 7
: 1153      1145      4
: 1154      1146      4      END
: 1155      1147      4
: 1156      1148      2      END;
: 1157      1149      2
: 1158      1150      2      ! Now calculate the number of blocks needed.
: 1159      1151      2
: 1160      1152      2      ! If UDF, ask for one block extra for overlapping of the buffers
: 1161      1153      2
: 1162      1154      2      IF .CONVSAB_IN_FAB [ FAB$B_RFM ] EQLU FAB$C_UDF
: 1163      1155      2      THEN
: 1164      1156      2      OUT_EXTRA = BLOCK_SIZE
: 1165      1157      2      ELSE
: 1166      1158      2      OUT_EXTRA = 0;
: 1167      1159      2
: 1168      1160      2      BEGIN
: 1169      1161      2
: 1170      1162      2      LOCAL
: 1171      1163      2      BYTES,
: 1172      1164      2      VFC_OFFSET;
: 1173      1165      2
: 1174      1166      2      ! Determine which is larger and use that size for the Buffer Size
: 1175      1167      2      !
: 1176      1168      2      BYTES = MAX( BLOCK_SIZE , ! At least a page
: 1177      1169      2      ( .IN_MRS + .IN_VFC ), ! Input record size
: 1178      1170      2      ( .CONV$GW_OUT_MRS + .OUT_VFC + .OUT_EXTRA )); ! Output record size
: 1179      1171      2
: 1180      1172      2      ! If we are doing a fast load get some extra bytes to use at the beginning
: 1181      1173      2      ! of the record for control information
: 1182      1174      2
: 1183      1175      2      IF .CONV$GL_FAST
: 1184      1176      2      THEN
: 1185      1177      2      BYTES = .BYTES + MAX_REC_CTRL;
: 1186      1178      2
: 1187      1179      2      !
: 1188      1180      2      ! IF UDF input, round buffer up to next whole block
: 1189      1181      2
: 1190      1182      2      IF .CONVSAB_IN_FAB [ FAB$B_RFM ] EQLU FAB$C_UDF
: 1191      1183      2      THEN
: 1192      1184      2      BYTES = (.BYTES + 511) AND NOT 511;
: 1193      1185      2

```



```

: 1194      1186      | Create the Buffer from virtual memory
: 1195      1187
: 1196      1188      CONV$GL_REC_BUF_PTR = CONV$$GET_VM ( .BYTES );
: 1197      1189
: 1198      1190      | If we doing a fast load hide the extra bytes at the beginning of
: 1199      1191      | record.
: 1200      1192
: 1201      1193      IF .CONV$GL_FAST
: 1202      1194      THEN
: 1203      1195          CONV$GL_REC_BUF_PTR = .CONV$GL_REC_BUF_PTR + MAX_REC_CTRL;
: 1204      1196
: 1205      1197      | Set the VFC offset to the max of the two offsets
: 1206      1198      VFC_OFFSET = MAX( .IN_VFC,.OUT_VFC );
: 1207      1199
: 1208      1200      | Correct the pointers and set the max. record size
: 1209      1201
: 1210      1202      CONV$GL_VFC_BUF_PTR = .CONV$GL_REC_BUF_PTR;
: 1211      1203      CONV$GL_REC_BUF_PTR = .CONV$GL_VFC_BUF_PTR + .VFC_OFFSET;
: 1212      1204
: 1213      1205      CONV$GW_MAX_REC_SIZ = .BYTES - .VFC_OFFSET
: 1214      1206
: 1215      1207      END;
: 1216      1208
: 1217      1209      RETURN CONV$_SUCCESS
: 1218      1210
: 1219      1211
: 1220      1212      END;

```

				OFFC 00000	.ENTRY	CONV\$\$CREATE_BUFFER, Save R2,R3,R4,R5,R6,-	
						R7,R8,R9,R10,R11	: 0959
		59	0000G	CF 9E 00002	MOVAB	CONV\$GL_REC_BUF_PTR, R9	
		58	0000G	CF 9E 00007	MOVAB	CONV\$AB_IN_FAB+31, R8	
		57	0000G	CF 9E 0000C	MOVAB	CONV\$GW_OUT_MRS, R7	
		56	0000G	CF 9E 00011	MOVAB	CONV\$AB_OUT_FAB+31, R6	
		03		66 91 00016	CMPB	CONV\$AB_OUT_FAB+31, #3	: 1022
				06 12 00019	BNEQ	1\$	
		54	20	A6 9A 0001B	MOVZBL	CONV\$AB_OUT_FAB+63, OUT_VFC	: 1024
				02 11 0001F	BRB	2\$	
				54 D4 00021	CLRL	OUT_VFC	: 1026
		67	17	A6 B0 00023	MOVW	CONV\$AB_OUT_FAB+54, CONV\$GW_OUT_MRS	: 1032
				42 12 00027	BNEQ	6\$	
		50	21	A6 D0 00029	MOVL	CONV\$AB_OUT_FAB+64, OUT_DEV	: 1041
0D		50		05 E1 0002D	BBC	#5, OUT_DEV, 3\$: 1043
		50	1D	A6 3C 00031	MOVZWL	CONV\$AB_OUT_FAB+60, R0	: 1045
		50		54 C2 00035	SUBL2	OUT_VFC, R0	
67		50		02 A3 00038	SUBW3	#2, R0, CONV\$GW_OUT_MRS	
				2D 11 0003C	BRB	6\$	
		51	FE	A6 9A 0003E	MOVZBL	CONV\$AB_OUT_FAB+29, R1	: 1049
				0D 12 00042	BNEQ	4\$	
08	FF	A6		03 E1 00044	BBC	#3, CONV\$AB_OUT_FAB+30, 4\$: 1050
67	01FE	8F		54 A3 00049	SUBW3	OUT_VFC, #50, CONV\$GW_OUT_MRS	: 1052
				1A 11 0004F	BRB	6\$	
		50	1F	A6 9A 00051	MOVZBL	CONV\$AB_OUT_FAB+62, R0	: 1058

50	50	09	78	00055	ASHL	#9, R0, R0	
	50	54	C2	00059	SUBL2	OUT_VFC, R0	1059
	10	51	91	0005C	CMPB	R1, #16	1056
		06	12	0005F	BNEQ	5\$	
67	50	03	A3	00061	SUBW3	#3, R0, CONV\$GW_OUT_MRS	1059
		04	11	00065	BRB	6\$	1058
67	50	07	A3	00067	SUBW3	#7, R0, CONV\$GW_OUT_MRS	1065
	51	68	9A	0006B	MOVZBL	CONVSAB_IN_FAB+31, R1	1072
		55	D4	0006E	CLRL	R5	
		51	D5	00070	TSTL	R1	
		2C	12	00072	BNEQ	9\$	
		55	D6	00074	INCL	R5	
	50	8F	3C	00076	MOVZWL	#512, IN_MRS	1076
	01	66	91	0007B	CMPB	CONVSAB_OUT_FAB+31, #1	1081
		08	12	0007E	BNEQ	7\$	
	0000G	CF	A6	B0	MOVW	CONVSAB_OUT_FAB+54, CONV\$GW_UDF_MRS	1083
		63	11	00086	BRB	15\$	
	0200	8F	B1	00088	CMPW	CONVS\$GW_OUT_MRS, #512	1088
		08	1E	0008D	BGEQU	8\$	
	50	00000000G	8F	D0	MOVL	#CONVS_UDF_BLK, R0	1090
			04	00096	RET		
	0000G	CF	8F	B0	MOVW	#512, CONV\$GW_UDF_MRS	1092
			4B	11	BRB	15\$	1081
	03		51	91	CMPB	R1, #3	1102
			06	12	BNEQ	10\$	
	52	20	A8	9A	MOVZBL	CONVSAB_IN_FAB+63, IN_VFC	1104
			02	11	BRB	11\$	
			52	D4	CLRL	IN_VFC	1106
	50	17	A8	3C	MOVZWL	CONVSAB_IN_FAB+54, IN_MRS	1111
			38	12	BNEQ	15\$	
	51	21	A8	D0	MOVL	CONVSAB_IN_FAB+64, IN_DEV	1120
09	51		05	E1	BBC	#5, IN_DEV, 12\$	1122
	50	1D	A8	3C	MOVZWL	CONVSAB_IN_FAB+60, IN_MRS	1124
	50		52	C2	SUBL2	IN_VFC, IN_MRS	
			27	11	BRB	15\$	
	53	FE	A8	9A	MOVZBL	CONVSAB_IN_FAB+29, R3	1129
			07	12	BNEQ	13\$	
	50	0000G	CF	3C	MOVZWL	CONVSAB_IN_XABFHC+10, IN_MRS	1131
			1A	11	BRB	15\$	
	51	1F	A8	9A	MOVZBL	CONVSAB_IN_FAB+62, R1	1137
51	51		09	78	ASHL	#9, R1, R1	
	51		52	C2	SUBL2	IN_VFC, R1	1138
	10		53	91	CMPB	R3, #16	1135
			06	12	BNEQ	14\$	
	50	FD	A1	9E	MOVAB	-3(R1), IN_MRS	1138
			04	11	BRB	15\$	1137
	50	F9	A1	9E	MOVAB	-7(R1), IN_MRS	1144
	07		55	E9	BLBC	R5, 16\$	1154
	53	0200	8F	3C	MOVZWL	#512, OUT_EXTRA	1156
			02	11	BRB	17\$	
			53	D4	CLRL	OUT_EXTRA	1158
	50		52	C0	ADDL2	IN_VFC, R0	1169
	51		67	3C	MOVZWL	CONVS\$GW_OUT_MRS, R1	1170
	51		54	C0	ADDL2	OUT_VFC, R1	
	51		53	C0	ADDL2	OUT_EXTRA, R1	
	51		50	D1	CPL	R0, R1	
			03	18	BGEQ	18\$	

00000200	50	51	D0	00108	MOVL	R1, R0		
	8F	50	D1	0010B	18\$:	CMPL	R0, #512	1168
		05	18	00112	BGEQ	19\$		
	50	0200	8F	3C	00114	MOVZWL	#512, R0	
	51		50	D0	00117	19\$:	MOVL	R0, BYTES
	03	0000G	CF	E9	0011C	BLBC	CONV\$GL_FAST, 20\$	1175
	51		0E	C0	00121	ADDL2	#14, BYTES	1177
	0D		55	E9	00124	20\$:	BLBC	R5, 21\$
	50	01FF	C1	9E	00127	MOVAB	511(R1), R0	1184
51	50	000001FF	8F	CB	0012C	BICL3	#511, R0, BYTES	
			51	DD	00134	21\$:	PUSHL	BYTES
			0000G	30	00136	BSBW	CONV\$\$GET_VM	1188
	5E		04	C0	00139	ADDL2	#4, SP	
	69		5C	0	0013C	MOVL	R0, CONV\$GL_REC_BUF_PTR	
	03	0000G	CF	9	0013F	BLBC	CONV\$GL_FAST, 22\$	1193
	69		0E	C0	00144	ADDL2	#14, CONV\$GL_REC_BUF_PTR	1195
	54		52	D1	00147	22\$:	CMPL	R2, OUT_VFC
			03	18	0014A	BGEQ	23\$	
	52		54	D0	0014C	MOVL	OUT_VFC, R2	
	50		52	D0	0014F	23\$:	MOVL	R2, VFC_OFFSET
	0000G		CF	69	00152	MOVL	CONV\$GL_REC_BUF_PTR, CONV\$GL_VFC_BUF_PTR	1203
	69	0000GDF	40	9E	00157	MOVAB	@CONV\$GL_VFC_BUF_PTR[VFC_OFFSET], -	1204
							CONV\$GL_REC_BUF_PTR	
	0000G	CF	51	A3	0015D	SUBW3	VFC_OFFSET, BYTES, CONV\$GW_MAX_REC_SIZ	1206
			50	D0	00163	MOVL	#1, R0	1210
				04	00166	RET		1212

: Routine Size: 359 bytes, Routine Base: _CONV\$CODE + 03D0

: 1221 1213 1
: 1222 1214 0 END ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
_CONV\$GLOBAL	8	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_CONV\$CODE	1335	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32:1	18619	96	0	1000	00:01.7
_\$255\$DUA28:[CONV.SRC]CONVERT.L32:1	165	14	8	17	00:00.2

CONVFILES
V04-000

VAX-11 CONVERT
CREATE_BUFFER

C 9
15-Sep-1984 23:45:35
14-Sep-1984 12:13:55

VAX-11 Bliss-32 V4.0-742
[CONV.SRC]CONVFILES.B32;1

Page 34
(9)

```
:  
:          COMMAND QUALIFIERS  
:  
:  BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:CONVFILES/OBJ=OBJ$:CONVFILES MSRC$:CONVFILES/UPDATE=(ENH$:CONVFILES)  
:  
: Size:          1335 code + 8 data bytes  
: Run Time:      00:29.5  
: Elapsed Time: 01:20.5  
: Lines/CPU Min: 2473  
: Lexemes/CPU-Min: 18442  
: Memory Used:  204 pages  
: Compilation Complete
```

The image displays a grid of 100 small, illegible document thumbnails arranged in 10 rows and 10 columns. The thumbnails are arranged in a regular grid pattern. Several thumbnails contain legible text labels, including:

- CONVDATA LIS
- CONVFSTLD LIS
- CONVFSTIO LIS
- CONVFSTRC LIS
- CONVFILES LIS
- CONVERROR LIS
- CONVFASFM LIS
- CONVDCL LIS

The remaining thumbnails are too small and faded to read, but they appear to contain various types of data, including lists, tables, and code snippets.