


```
CCCCCCCC 000000 NN NN VV VV FFFFFFFF SSSSSSSS TTTTTTTTTT IIIIII 000000
CCCCCCCC 000000 NN NN VV VV FFFFFFFF SSSSSSSS TTTTTTTTTT IIIIII 000000
CC CC 00 00 NN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NNNN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NNNN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FFFFFFFF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FFFFFFFF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FF FF SS SS TT TT II II 00 00
CC CC 00 00 NN NN NN VV VV FF FF SS SS TT TT II II 00 00
CCCCCCCC 000000 NN NN VV VV FF FF SS SS TT TT IIIIII 000000
CCCCCCCC 000000 NN NN VV VV FF FF SS SS TT TT IIIIII 000000

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



```
0001 0 %TITLE 'VAX-11 CONVERT'
0002 0 MODULE CONV$FSTIO ( 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: CONVERT fast load I/O and bucket routines
36 0035 1
37 0036 1 Contents:
38 0037 1 WRITE_BUCKET
39 0038 1 SET_NXTBKT
40 0039 1 SWAP_BUCKET
41 0040 1 GET_BUCKET
42 0041 1 INIT_BUCKET
43 0042 1 EXTEND_AREA
44 0043 1 CONVERT_VBN_ID
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
52 0051 1
53 0052 1 Author: Keith B Thompson Creation date: August-1980
54 0053 1
55 0054 1
56 0055 1 Modified by:
57 0056 1
58 0057 1 V03-005 RAS0331 Ron Schaefer 31-Jul-1984
59 0058 1 Accumulate total area allocation.
60 0059 1
61 0060 1 V03-004 KBT0478 Keith B. Thompson 29-Jan-1983
62 0061 1 Make extend_area global
63 0062 1
64 0063 1 V03-003 KBT0385 Keith B. Thompson 27-Oct-1982
65 0064 1 Make changes to support prologue 3 sidrs
66 0065 1
67 0066 1 V03-002 KBT0350 Keith B. Thompson 4-Oct-1982
68 0067 1 Use new linkage definitions
69 0068 1
70 0069 1 V03-001 KBT0024 Keith Thompson 25-Mar-1982
71 0070 1 Change the linkage to get_bucket and remove some useless code
72 0071 1
73 0072 1 ****
```



```

: 75      0073 1
: 76      0074 1 PSECT
: 77      0075 1      OWN      = _CONVF$FAST_D (PIC),
: 78      0076 1      GLOBAL   = _CONVF$FAST_D (PIC),
: 79      0077 1      PLIT     = _CONVF$PLIT_ (SHARE,PIC),
: 80      0078 1      CODE     = _CONVF$FAST_S (SHARE,PIC);
: 81      0079 1
: 82      0080 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 83      0081 1 LIBRARY 'SRC$:CONVERT';
: 84      0082 1
: 85      0083 1 DEFINE_ERROR_CODES;
: 86      0084 1
: 87      0085 1 EXTERNAL ROUTINE
: 88      0086 1      CONVF$$GET_TEMP_VM      : CL$GET_TEMP_VM,
: 89      0087 1      CONVF$$RMS_ERROR        : NOVALUE,
: 90      0088 1      CONVF$$RMS_READ_ERROR    : NOVALUE;
: 91      0089 1
: 92      0090 1 FORWARD ROUTINE
: 93      0091 1      SET_NXTBKT              : CL$JSB_REG_9 NOVALUE,
: 94      0092 1      SWAP_BUCKET              : CL$JSB_REG_9 NOVALUE,
: 95      0093 1      CONVF$$INIT_BUCKET        : CL$JSB_REG_9 NOVALUE,
: 96      0094 1      CONVF$$EXTEND_AREA        : CL$EXTEND_AREA NOVALUE;
: 97      0095 1
: 98      0096 1 EXTERNAL
: 99      0097 1      CONVF$GL_RFA_BUFFER,
100      0098 1
101      0099 1      CONVF$AB_OUT_FAB           : $FAB_DECL,
102      0100 1      CONVF$AB_OUT_RAB           : $RAB_DECL,
103      0101 1
104      0102 1      CONVF$GL_CTX_BLOCK,
105      0103 1
106      0104 1      CONVF$GL_EOF_VBN,
107      0105 1      CONVF$GB_PROG_V3           : BYTE,
108      0106 1      CONVF$AR_AREA_BLOCK        : REF BLOCKVECTOR [ ,AREASC_BLN,BYTE ],
109      0107 1      CONVF$GW_AREA_SIZE        : WORD;
110      0108 1
111      0109 1 MACRO
112      0110 1
113      0111 1      ! These macros make the code look a little better
114      0112 1      !
115      0113 1      BKT$W_VBNFS      = .CONVF$GW_VBN_FS_PTR,0,16,0% ! VBN Freespace Pointer in index level
116      0114 1      BKT$W_VBNFS0    = .CONVF$GW_VBN_FS_PTR0,0,16,0% ! VBN Freespace Pointer at the data level
117      0115 1      BKT$L_LCBPTR    = .CONVF$GW_LCB_PTR,0,32,0% ! Last Continuation Bucket Pointer
118      0116 1
119      0117 1 EXTERNAL
120      0118 1      CONVF$GW_VBN_FS_PTR      : WORD,
121      0119 1      CONVF$GW_VBN_FS_PTR0     : WORD,
122      0120 1      CONVF$GW_LCB_PTR        : WORD;
123      0121 1
124      0122 1 GLOBAL CONVF$GL_CONT_VBN      : LONG;
125      0123 1
```

```
127 0124 1 %SBTTL 'WRITE_BUCKET'
128 0125 1 GLOBAL ROUTINE CONV$$WRITE_BUCKET : CL$JSB_REG_9 NOVALUE =
129 0126 1 ++
130 0127 1
131 0128 1 Functional Description:
132 0129 1
133 0130 1 Writes the current bucket into the output file
134 0131 1
135 0132 1 Calling Sequence:
136 0133 1
137 0134 1 CONV$$WRITE_BUCKET()
138 0135 1
139 0136 1 Input Parameters:
140 0137 1 none
141 0138 1
142 0139 1 Implicit Inputs:
143 0140 1 none
144 0141 1
145 0142 1 Output Parameters:
146 0143 1 none
147 0144 1
148 0145 1 Implicit Outputs:
149 0146 1 none
150 0147 1
151 0148 1 Routine Value:
152 0149 1 none
153 0150 1
154 0151 1 Routines Called:
155 0152 1
156 0153 1 SET_NXTBKT
157 0154 1 SWAP_BUCKET
158 0155 1 $WRITE
159 0156 1 CONV$$RMS_ERROR - By RMS as an AST
160 0157 1
161 0158 1 Side Effects:
162 0159 1 none
163 0160 1
164 0161 1 --
165 0162 1
166 0163 2 BEGIN
167 0164 2
168 0165 2 DEFINE_CTX;
169 0166 2 DEFINE_BUCKET;
170 0167 2 DEFINE_KEY_DESC;
171 0168 2
172 0169 2 ! Set the next bucket pointer to the bucket
173 0170 2 !
174 0171 2 SET_NXTBKT();
175 0172 2
176 0173 2 ! Point RMS to the bucket. NOTE: This is where OUT_RAB is changed
177 0174 2 !
178 0175 2 CONV$AB_OUT_RAB [ RAB$L_RBF ] = .BUCKET;
179 0176 2 CONV$AB_OUT_RAB [ RAB$W_RSZ ] = .CTX [ CTX$L_SIZ ];
180 0177 2 CONV$AB_OUT_RAB [ RAB$L_BKT ] = .CTX [ CTX$L_CURRENT_VBN ];
181 0178 2
182 0179 2 ! If we are doing double buffering on this bucket
183 0180 2 ! then swap pointers to the buckets and set for asyc. operation
```



```
184 0181 2 !
185 0182 ! IF .CTX [ CTX$V_DBF ]
186 0183 THEN
187 0184 BEGIN
188 0185 ! Switch the buffer
189 0186 !
190 0187 SWAP_BUCKET();
191 0188 !
192 0189 ! Set the asynchronous bit
193 0190 !
194 0191 CONV$AB_OUT_RAB [ RAB$V_ASY ] = _SET
195 0192 !
196 0193 END
197 0194 ELSE
198 0195 ! If not then we need a syncrous call
199 0196 !
200 0197 CONV$AB_OUT_RAB [ RAB$V_ASY ] = _CLEAR;
201 0198 !
202 0199 ! Wait on the last IO if necessary
203 0200 !
204 0201 $WAIT ( RAB=CONV$AB_OUT_RAB );
205 0202 !
206 0203 ! Write The Bucket
207 0204 !
208 0205 $WRITE ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_ERROR );
209 0206 !
210 0207 RETURN
211 0208 !
212 0209 !
213 0210 !
214 0211 1 END;
```

```
.TITLE CONV$FSTIO VAX-11 CONVERT
.IDENT \V04-000\
```

```
.PSECT _CONV$FAST_D,NOEXE, PIC,2
```

```
00000 CONV$GL_CONT VBN::
.BLK 4
```

```
.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$$GET_TEMP_VM
.EXTRN CONV$$RMS_ERROR
.EXTRN CONV$$RMS_READ_ERROR
.EXTRN CONV$GL_RFA_BUFFER
.EXTRN CONV$AB_OUT_FAB
.EXTRN CONV$AB_OUT_RAB
.EXTRN CONV$GL_CTX_BLOCK
.EXTRN CONV$GL_EOF_VBN
.EXTRN CONV$GB_PROL_V3
.EXTRN CONV$AR_AREA_BLOCK
.EXTRN CONV$GW_AREA_SIZE
.EXTRN CONV$GW_VBN_FS_PTR
.EXTRN CONV$GW_VBN_FS_PTRO
.EXTRN CONV$GW_LCB_PTR
.EXTRN SYSS$WAIT, SYSS$WRITE
```

```
.PSECT _CONVSFAST_S, NOWRT, SHR, PIC, 2
```

			0000V 30 00000	CONV\$\$WRITE_BUCKET::		
				BSBW	SET NXTBKT	: 0171
	0000G	CF	59 D0 00003	MOVL	BUCKET, CONV\$AB_OUT_RAB+40	: 0175
	0000G	CF	20 AA B0 00008	MOVW	32(CTX), CONV\$AB_OUT_RAB+34	: 0176
	0000G	CF	08 AA D0 0000E	MOVL	8(CTX), CONV\$AB_OUT_RAB+56	: 0177
0A		6A	03 E1 00014	BBC	#3, (CTX), 1\$	: 0182
			0000V 30 00018	BSBW	SWAP_BUCKET	: 0188
	0000G	CF	01 88 0001B	BISB2	#1, CONV\$AB_OUT_RAB+4	: 0192
			05 11 00020	BRB	2\$	: 0199
	0000G	CF	01 8A 00022 1\$:	BICB2	#1, CONV\$AB_OUT_RAB+4	: 0203
	00000000G	00	0000G CF 9F 00027 2\$:	PUSHAB	CONV\$AB_OUT_RAB	: 0207
			0000G CF 9F 0002B	CALLS	#1, SYSS\$WAIT	: 0211
	00000000G	00	0000G CF 9F 00032	PUSHAB	CONV\$\$RMS_ERROR	
			0000G CF 9F 00036	PUSHAB	CONV\$AB_OUT_RAB	
			02 FB 0003A	CALLS	#2, SYSS\$WRITE	
			05 00041	RSB		

; Routine Size: 66 bytes, Routine Base: _CONVSFAST_S + 0000

; 215 0212 1


```
217 0213 1 %SBTTL 'SET_NXTBKT'
218 0214 1 ROUTINE SET_NXTBKT : CL$JSB_REG_9 NOVALUE =
219 0215 1 ++
220 0216 1
221 0217 1 Functional Description:
222 0218 1
223 0219 1 Writes the next bucket vbn field in the current bucket
224 0220 1
225 0221 1 Calling Sequence:
226 0222 1
227 0223 1 SET_NXTBKT()
228 0224 1
229 0225 1 Input Parameters:
230 0226 1 none
231 0227 1
232 0228 1 Implicit Inputs:
233 0229 1 none
234 0230 1
235 0231 1 Output Parameters:
236 0232 1 none
237 0233 1
238 0234 1 Implicit Outputs:
239 0235 1 none
240 0236 1
241 0237 1 Routine Value:
242 0238 1 none
243 0239 1
244 0240 1 Routines Called:
245 0241 1 none
246 0242 1
247 0243 1 Side Effects:
248 0244 1 none
249 0245 1
250 0246 1 --
251 0247 1
252 0248 2 BEGIN
253 0249 2
254 0250 2 DEFINE_CTX;
255 0251 2 DEFINE_BUCKET;
256 0252 2
257 0253 2 LOCAL AREA;
258 0254 2
259 0255 2 ! Get the area that the bucket is in
260 0256 2
261 0257 2 AREA = .CTX [ CTX$B_AREA ];
262 0258 2
263 0259 2 ! If this is the last bucket in a horz. chain
264 0260 2 ! then write back pointers to the beginning of the chain
265 0261 2 ! else write next bucket pointers
266 0262 2
267 0263 2 IF .BUCKET [ BKT$V_LASTBKT ]
268 0264 2 THEN
269 0265 2
270 0266 2 ! Next bucket pointer points to the first bucket in this chain
271 0267 2
272 0268 2 BUCKET [ BKT$L_NXTBKT ] = .CTX [ CTX$L_FIRST_VBN ]
273 0269 2 ELSE
```

RETURN
END;

```
; Routine Size: 66 bytes,   Routine Base: _CONV$FAST_S + 0042
```



```
297 0292 1 %SBTTL 'SWAP_BUCKET'
298 0293 1 ROUTINE SWAP_BUCKET : CL$JSB_REG_9 NOVALUE =
299 0294 1 ++
300 0295 1
301 0296 1 Functional Description:
302 0297 1
303 0298 1 Swaps the current bucket with the secondary bucket for
304 0299 1 double buffering
305 0300 1
306 0301 1 Calling Sequence:
307 0302 1
308 0303 1 SWAP_BUCKET()
309 0304 1
310 0305 1 Input Parameters:
311 0306 1 none
312 0307 1
313 0308 1 Implicit Inputs:
314 0309 1 none
315 0310 1
316 0311 1 Output Parameters:
317 0312 1 none
318 0313 1
319 0314 1 Implicit Outputs:
320 0315 1 none
321 0316 1
322 0317 1 Routine Value:
323 0318 1 none
324 0319 1
325 0320 1 Routines Called:
326 0321 1 none
327 0322 1
328 0323 1 Side Effects:
329 0324 1 none
330 0325 1
331 0326 1 --
332 0327 1
333 0328 2 BEGIN
334 0329 2
335 0330 2 DEFINE_CTX;
336 0331 2 DEFINE_BUCKET;
337 0332 2
338 0333 2 IF .CTX [ CTX$V_DBX ]
339 0334 2 THEN
340 0335 2 BEGIN
341 0336 2 BUCKET = .CTX [ CTX$SL_PTO ];
342 0337 2 CTX [ CTX$SL_CURRENT_BUFFER ] = .CTX [ CTX$SL_PTO ];
343 0338 2 CTX [ CTX$SL_END ] = .CTX [ CTX$SL_ENO ]
344 0339 2 END
345 0340 2 ELSE
346 0341 2 BEGIN
347 0342 2 BUCKET = .CTX [ CTX$SL_PT1 ];
348 0343 2 CTX [ CTX$SL_CURRENT_BUFFER ] = .CTX [ CTX$SL_PT1 ];
349 0344 2 CTX [ CTX$SL_END ] = .CTX [ CTX$SL_EN1 ]
350 0345 2 END;
351 0346 2
352 0347 2 CTX [ CTX$V_DBX ] = NOT .CTX [ CTX$V_DBX ];
353 0348 2
```

CONV\$FST10
V04-000

VAX-11 CONVERT
SWAP_BUCKET

N 9
15-Sep-1984 23:54:11
14-Sep-1984 12:13:57

VAX-11 Bliss-32 V4.0-742
[CONV.SRC]CONVFST10.B32;1

Page 10
(6)

: 354
: 355
: 356
0349 2 RETURN
0350 2
0351 1 END;

10

6A

04 E1 00000 SWAP_BUCKET:

	59	10	AA	D0	00004	BBC	#4, (CTX), 1\$	
04	AA	10	AA	D0	00008	MOVL	16(CTX), BUCKET	
0C	AA	14	AA	D0	0000D	MOVL	16(CTX), 4(CTX)	
			0E	11	00012	MOVL	20(CTX), 12(CTX)	
	59	18	AA	D0	00014	BRB	2\$	
04	AA	18	AA	D0	00018	MOVL	24(CTX), BUCKET	
0C	AA	1C	AA	D0	0001D	MOVL	24(CTX), 4(CTX)	
	6A		10	8C	00022	MOVL	28(CTX), 12(CTX)	
			05	00025	2\$:	XORB2	#16, (CTX)	
						RSB		

: 0333
: 0336
: 0337
: 0338
: 0342
: 0343
: 0344
: 0347
: 0351

; Routine Size: 38 bytes, Routine Base: _CONV\$FAST_S + 0084


```
358 0352 1 %SBTTL 'GET_BUCKET'
359 0353 1 GLOBAL ROUTINE CONV$$GET_BUCKET ( AREA ) : CL$JSB_REG_9 NOVALUE =
360 0354 1 ++
361 0355 1
362 0356 1 Functional Description:
363 0357 1
364 0358 1     Allocates and formats a pair of buckets in memory
365 0359 1
366 0360 1 Calling Sequence:
367 0361 1
368 0362 1     CONV$$GET_BUCKET( area )
369 0363 1
370 0364 1 Input Parameters:
371 0365 1
372 0366 1     area      - Area from witch the bucket is to be allocated
373 0367 1
374 0368 1 Implicit Inputs:
375 0369 1     none
376 0370 1
377 0371 1 Output Parameters:
378 0372 1     none
379 0373 1
380 0374 1 Implicit Outputs:
381 0375 1     none
382 0376 1
383 0377 1 Routine Value:
384 0378 1     none
385 0379 1
386 0380 1 Routines Called:
387 0381 1
388 0382 1     CONV$$GET_TEMP_VM
389 0383 1     INIT_BUCKET
390 0384 1
391 0385 1 Side Effects:
392 0386 1     none
393 0387 1
394 0388 1 --
395 0389 1
396 0390 2 BEGIN
397 0391 2
398 0392 2 DEFINE_CTX;
399 0393 2 DEFINE_BUCKET;
400 0394 2 DEFINE_KEY_DESC;
401 0395 2
402 0396 2 LOCAL
403 0397 2     BYTES;
404 0398 2
405 0399 2     ! Get the number of bytes per bucket for that area:
406 0400 2     ! ( bytes = blocks * block_size * 2 ) NOTE: *2 for double buffering
407 0401 2     !
408 0402 2     BYTES = ( .CONV$AR_AREA_BLOCK [ .AREA, AREA$B_ARBKTSZ ] * BLOCK_SIZE * 2 );
409 0403 2
410 0404 2     ! Get the space.
411 0405 2     !
412 0406 2     BUCKET = CONV$$GET_TEMP_VM ( .BYTES );
413 0407 2
414 0408 2     ! For double buffering then hide the extra buffer for later
```



```

415      0409 2      !
416      0410 2      BYTES = .BYTES / 2;
417      0411 2
418      0412 2      ! Initialize some static values
419      0413 2
420      0414 2      CTX [ CTX$$_CURRENT_BUFFER ] = .BUCKET;
421      0415 2      CTX [ CTX$$_SIZ ] = .BYTES;
422      0416 2
423      0417 2      ! CTX$$_END points at the first free byte BEFORE the check byte
424      0418 2
425      0419 2      CTX [ CTX$$_END ] = .BUCKET + .BYTES - 2;
426      0420 2      CTX [ CTX$$_AREA ] = .AREA;
427      0421 2
428      0422 2      ++
429      0423 2
430      0424 2      Init. static fields in the bucket
431      0425 2
432      0426 2      --
433      0427 2
434      0428 2      ! Level (all prologues)
435      0429 2
436      0430 2      BUCKET [ BKT$$_LEVEL ] = .CTX [ CTX$$_LEVEL ];
437      0431 2
438      0432 2      ! Prologue dependent fields and some pointers into the
439      0433 2      bucket and some sizes
440      0434 2
441      0435 2      IF .CONV$$_PROL_V3
442      0436 2      THEN
443      0437 2          BEGIN          ! Prologue 3
444      0438 2
445      0439 2          ! Bucket key of ref
446      0440 2
447      0441 2          BUCKET [ BKT$$_INDEXNO ] = .KEY_DESC [ KEY$$_KEYREF ];
448      0442 2
449      0443 2          ! For level 0 (data) buckets we can have a LCB pointer
450      0444 2
451      0445 2          IF .CTX [ CTX$$_LEVEL ] EQL 0
452      0446 2          THEN
453      0447 2
454      0448 2              IF ( .KEY_DESC [ KEY$$_KEYREF ] EQL 0 ) AND
455      0449 2                  .KEY_DESC [ KEY$$_DUPKEYS ]
456      0450 2              THEN
457      0451 2                  BEGIN
458      0452 2
459      0453 2                  ! Only primary data bucket have a LCB pointer
460      0454 2
461      0455 2                  CONV$$_LCB_PTR = .BYTES - 6;
462      0456 2
463      0457 2                  CTX [ CTX$$_FREE ] = .BYTES - BKT$$_OVERHDSZ - 6
464      0458 2
465      0459 2                  END
466      0460 2              ELSE
467      0461 2                  CTX [ CTX$$_FREE ] = .BYTES - BKT$$_OVERHDSZ - 2
468      0462 2
469      0463 2          ELSE
470      0464 2              BEGIN
471      0465 2
```



```

472      0466 4      ! Index buckets only have a VBN freespace pointer
473      0467 4
474      0468 4      CONV$GW_VBN_FS_PTR = .BYTES - 4;
475      0469 4
476      0470 4      CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 4
477      0471 4
478      0472 4      END
479      0473 4      END      ! Prologue 3
480      0474 4      ELSE
481      0475 4      BEGIN      ! Prologue 1,2
482      0476 4
483      0477 4      ! Bucket area number
484      0478 4
485      0479 4      BUCKET [ BKT$B_AREANO ] = .AREA;
486      0480 4
487      0481 4      ! Highest record id
488      0482 4
489      0483 4      BUCKET [ BKT$B_LSTRECID ] = 255;
490      0484 4
491      0485 4      ! The space avail. is bytes - overhead - check byte
492      0486 4
493      0487 4      CTX [ CTX$W_FREE ] = .BYTES - BKT$C_OVERHDSZ - 1
494      0488 4
495      0489 4      END;      ! Prologue 1,2
496      0490 4
497      0491 4      ! For double buffering set up the pointers to the buffers and init
498      0492 4      the extra buffer
499      0493 4
500      0494 4      Set up the various pointers
501      0495 4
502      0496 4      CTX [ CTX$L_PTO ] = .BUCKET;
503      0497 4      CTX [ CTX$L_PT1 ] = .BUCKET + .BYTES;
504      0498 4      CTX [ CTX$L_ENO ] = .CTX [ CTX$L_END ];
505      0499 4      CTX [ CTX$L_EN1 ] = .CTX [ CTX$L_END ] + .BYTES;
506      0500 4
507      0501 4      ! Init the second buffer by coping the static overhead into it
508      0502 4
509      0503 4      CH$MOVE( BKT$K_OVERHDSZ + 1, .CTX [ CTX$L_PTO ], .CTX [ CTX$L_PT1 ] );
510      0504 4
511      0505 4      ! Tell everyone that we are doind double buffering and which bucket
512      0506 4      are pointing to
513      0507 4
514      0508 4      CTX [ CTX$V_DBF ] = _SET;
515      0509 4      CTX [ CTX$V_DBX ] = _CLEAR;      ! Clear = bucket 0, Set = bucket 1
516      0510 4
517      0511 4      ! Initialize dynamic values and update area descriptor
518      0512 4
519      0513 4      CONV$$INIT_BUCKET();
520      0514 4
521      0515 4      ! Set the pointer for this level
522      0516 4
523      0517 4      CTX [ CTX$L_FIRST_VBN ] = .CTX [ CTX$L_CURRENT_VBN ];
524      0518 4
525      0519 4      ! Say that the bucket is ready and that will have the first record
526      0520 4
527      0521 4      CTX [ CTX$V_RDY ] = _SET;
528      0522 4      CTX [ CTX$V_FST ] = _SET;
```

: 529
: 530
: 531
: 532

0523 2
0524 2
0525 2
0526 1

RETURN
END;

```

                                3C BB 00000 CONV$$GET_BUCKET::
                                POSHR #^M<R2,R3,R4,R5>
50      14 AE 06 78 00002 ASHL #6, AREA, R0
                                ADDL2 CONV$AR AREA_BLOCK, R0
51      51 03 0A 9A 0000C MOVZBL 3(R0), BYTES
                                ASHL #10, BYTES, BYTES
                                PUSHL BYTES
                                BSBW CONV$$GET_TEMP_VM
                                ADDL2 #4, SP
                                MOVL R0, BUCKET
                                DIVL2 #2, BYTES
                                MOVL BUCKET, 4(CTX)
                                MOVL BYTES, 32(CTX)
                                MOVAB -2(BYTES)[BUCKET], 12(CTX)
                                MOVB AREA, 1(CTX)
                                MOVB 2(CTX), 12(BUCKET)
                                MOVAB -14(R1), R0
                                BLBC CONV$GB_PROL_V3, 3$
                                MOVB 21(KEY_DESC), 1(BUCKET)
                                TSTB 2(CTX)
                                BNEQ 2$
                                TSTB 21(KEY_DESC)
                                BNEQ 1$
                                BLBC 16(KEY_DESC), 1$
                                SUBW3 #6, BYTES, CONV$GW_LCB_PTR
                                SUBW3 #6, R0, 40(CTX)
                                BRB 4$
                                SUBW3 #2, R0, 40(CTX)
                                BRB 4$
                                SUBW3 #4, BYTES, CONV$GW_VBN_FS_PTR
                                SUBW3 #4, R0, 40(CTX)
                                BRB 4$
                                MOVB AREA, 1(BUCKET)
                                MNEGB #1, 7(BUCKET)
                                SUBW3 #1, R0, 40(CTX)
                                MOVL BUCKET, 16(CTX)
                                ADDL3 BYTES, BUCKET, 24(CTX)
                                MOVL 12(CTX), 20(CTX)
                                MOVAB @12(CTX)[BYTES], 28(CTX)
                                MOVC3 #15, @16(CTX), @24(CTX)
                                BISB2 #8, (CTX)
                                BICB2 #16, (CTX)
                                BSBW CONV$$INIT_BUCKET
                                MOVL 8(CTX), 36(CTX)
                                BISB2 #5, (CTX)
                                POPR #^M<R2,R3,R4,R5>
                                RSB

0000G CF 51 06 78 00002
28 AA 50 0A 9A 0000C
28 AA 51 0A 78 00010
0000G 51 DD 00014
04 50 04 C0 00019
59 50 D0 0001C
51 02 C6 0001F
04 AA 59 D0 00022
20 AA 51 D0 00026
0C AA FE A1 49 9E 0002A
01 AA 14 AE 90 00030
0C A9 02 AA 90 00035
50 F2 A1 9E 0003A
34 0000G CF E9 0003E
01 A9 15 AB 90 00043
02 AA 95 00048
1D 12 0004B
15 AB 95 0004D
11 12 00050
10 AB E9 00052
0000G CF 51 06 A3 00056
28 AA 50 06 A3 0005C
28 AA 50 22 11 00061
0000G CF 51 02 A3 00063 1$:
28 AA 50 1B 11 00068
0000G CF 51 04 A3 0006A 2$:
28 AA 50 04 A3 00070
01 A9 0E 11 00075
07 A9 14 AE 90 00077 3$:
28 AA 50 01 8E 0007C
10 AA 59 A3 00080
18 AA 51 C1 00089 4$:
14 AA 0C AA D0 0008E
1C AA 0C BA 41 9E 00093
18 BA 10 0F 28 00099
6A 08 88 0009F
6A 10 8A 000A2
0000V 30 000A5
24 AA 08 AA D0 000A8
6A 05 88 000AD
3C BA 000B0
05 05 000B2
```


CONVFST10
V04-000

VAX-11 CONVERT
GET_BUCKET

F 10
15-Sep-1984 23:54:11
14-Sep-1984 12:13:57

VAX-11 Bliss-32 V4.0-742
[CONV.SRC]CONVFST10.B32;1

Page 15
(7)

; Routine Size: 179 bytes, Routine Base: _CONVFST10_

```
534 0527 1 %SBTTL 'INIT_BUCKET'
535 0528 1 GLOBAL ROUTINE CONVS$INIT_BUCKET : CL$JSB_REG_9 NOVALUE =
536 0529 1 ++
537 0530 1
538 0531 1 Functional Description:
539 0532 1
540 0533 1 Gets a new VBN for a bucket in the proper area and initializes
541 0534 1 all of the dynamic fields in the bucket
542 0535 1
543 0536 1 Calling Sequence:
544 0537 1
545 0538 1 CONVS$INIT_BUCKET()
546 0539 1
547 0540 1 Input Parameters:
548 0541 1 none
549 0542 1
550 0543 1 Implicit Inputs:
551 0544 1 none
552 0545 1
553 0546 1 Output Parameters:
554 0547 1 none
555 0548 1
556 0549 1 Implicit Outputs:
557 0550 1 none
558 0551 1
559 0552 1 Routine Value:
560 0553 1 none
561 0554 1
562 0555 1 Routines Called:
563 0556 1
564 0557 1 CONVS$EXTEND_AREA
565 0558 1
566 0559 1 Side Effects:
567 0560 1
568 0561 1 Could extend the allocation of the output file
569 0562 1
570 0563 1 --
571 0564 1
572 0565 2 BEGIN
573 0566 2
574 0567 2 DEFINE_CTX;
575 0568 2 DEFINE_BUCKET;
576 0569 2 DEFINE_KEY_DESC;
577 0570 2
578 0571 2 LOCAL AREA;
579 0572 2
580 0573 2 AREA = .CTX [ CTX$B_AREA ];
581 0574 2
582 0575 2 ! See if the bucket will fit in the current extent if it doesnt extend the
583 0576 2 ! file.
584 0577 2
585 0578 3 IF ( .CONVS$AR_BLOCK [ .AREA, AREA$B_ARBKTSZ ] GTR
586 0579 4 ( .CONVS$AR_BLOCK [ .AREA, AREA$C_CNBLK ] -
587 0580 3 .CONVS$AR_BLOCK [ .AREA, AREA$L_USED ] ) )
588 0581 2 THEN
589 0582 2
590 0583 2 ! Extend area
```



```

591      0584      2      !
592      0585      2      !CONVS$EXTEND_AREA ( .AREA );
593      0586      2
594      0587      2      ! Set the VBN of the bucket and determine the size of the VBN pointers
595      0588      2
596      0589      2      BEGIN          ! VBN local
597      0590      2
598      0591      2      LOCAL          VBN;
599      0592      2
600      0593      2      ! Get the next VBN of this bucket
601      0594      2
602      0595      2      VBN = .CONVSAR_AREA_BLOCK [ .AREA,AREASL_NXTVBN ];
603      0596      2
604      0597      2      CTX [ CTX$SL_CURRENT_VBN ] = .VBN;
605      0598      2
606      0599      2      ! Determine the pointer size needed for this VBN
607      0600      2
608      0601      2      IF .VBN LSS 65536
609      0602      2      THEN
610      0603      2          CTX [ CTX$V_VBN ] = 0          ! 2 byte pointer
611      0604      2      ELSE
612      0605      2          IF .VBN LSS 1048576
613      0606      2          THEN
614      0607      2              CTX [ CTX$V_VBN ] = 1          ! 3 byte
615      0608      2          ELSE
616      0609      2              CTX [ CTX$V_VBN ] = 2          ! 4 byte
617      0610      2
618      0611      2      END;          ! VBN local
619      0612      2
620      0613      2      !++
621      0614      2      !
622      0615      2      ! Update the area descriptor to account for the new bucket
623      0616      2      !
624      0617      2      !--
625      0618      2
626      0619      2      ! Correct the pointers and counters in the prologue area descriptor
627      0620      2
628      0621      2      CONVSAR_AREA_BLOCK [ .AREA,AREASL_USED ] =
629      0622      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASL_USED ] +
630      0623      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASB_ARBKTSZ ];
631      0624      2
632      0625      2      CONVSAR_AREA_BLOCK [ .AREA,AREASL_NXTVBN ] =
633      0626      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASL_NXTVBN ] +
634      0627      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASB_ARBKTSZ ];
635      0628      2
636      0629      2      CONVSAR_AREA_BLOCK [ .AREA,AREASL_TOTAL_ALLOC ] =
637      0630      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASL_TOTAL_ALLOC ] +
638      0631      2          .CONVSAR_AREA_BLOCK [ .AREA,AREASB_ARBKTSZ ];
639      0632      2
640      0633      2      !++
641      0634      2      !
642      0635      2      ! Init. dynamic fields in bucket
643      0636      2      !
644      0637      2      !--
645      0638      2
646      0639      2      ! Bucket control byte (all prologues)
647      0640      2      !

```

```

: 648      0641 2      BUCKET [ BKT$B_BKTCB ] = _CLEAR;
: 649      0642 2
: 650      0643 2      ! The freespace always points just past the bucket overhead (all prologues)
: 651      0644 2
: 652      0645 2      BUCKET [ BKT$W_FREESPACE ] = BKT$C_OVERHDSZ;
: 653      0646 2
: 654      0647 2      ! Set address sample (all prologues)
: 655      0648 2
: 656      0649 2      BUCKET [ BKT$W_ADRSAMPLE ] = .CTX [ CTX$L_CURRENT_VBN ];
: 657      0650 2
: 658      0651 2      ! Prologue dependent fields
: 659      0652 2
: 660      0653 2      IF .CONV$GB_PROL_V3
: 661      0654 2      THEN
: 662      0655 2          BEGIN
: 663      0656 2              ! Prologue 3
: 664      0657 2              ! Index buckets have VBN freespace pointers
: 665      0658 2              !
: 666      0659 2              IF .CTX [ CTX$B_LEVEL ] NEQ 0
: 667      0660 2              THEN
: 668      0661 4                  BEGIN
: 669      0662 4                      LOCAL CTX_M1 : REF BLOCK [ ,BYTE ];
: 670      0663 4
: 671      0664 4                      ! The VBN of the bucket one level down determines size
: 672      0665 4                      ! of the VBN pointers in this bucket
: 673      0666 4                      !
: 674      0667 4                      CTX_M1 = .CTX - CTX$K_BLN;
: 675      0668 4                      BUCKET [ BKT$V_PTR_SZ ] = .CTX_M1 [ CTX$V_VBN ];
: 676      0669 4
: 677      0670 4                      ! The vbn freespace points to the byte just before the pointer
: 678      0671 4                      !
: 679      0672 4                      BUCKET [ BKT$W_VBNFS ] = .CONV$GW_VBN_FS_PTR - 1
: 680      0673 4
: 681      0674 4                      END;
: 682      0675 4
: 683      0676 3              ! Reset the next record ID
: 684      0677 3              BUCKET [ BKT$W_NXTRECID ] = 1
: 685      0678 3
: 686      0679 3          END
: 687      0680 3              ! Prologue 3
: 688      0681 3      ELSE
: 689      0682 3          BEGIN
: 690      0683 3              ! Prologue 1,2
: 691      0684 3              ! Reset the record ID
: 692      0685 3              !
: 693      0686 3              BUCKET [ BKT$B_NXTRECID ] = 1
: 694      0687 3
: 695      0688 3              END;
: 696      0689 3              ! Prologue 1,2
: 697      0690 2
: 698      0691 2      ! Reset the available space in the bucket
: 699      0692 2      !
: 700      0693 2      CTX [ CTX$W_SPC ] = .CTX [ CTX$W_FREE ];
: 701      0694 2
: 702      0695 2      ! Indicate that the bucket has not been used yet
: 703      0696 2      !
: 704      0697 2
```


CONVSFSTIO
V04-000

VAX-11 CONVERT
INIT_BUCKET

J 10
15-Sep-1984 23:54:11
14-Sep-1984 12:13:57

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

Page 19
(8)

```
: 705      0698 2      CTX [ CTX$W_USE ] = 0;
: 706      0699 2
: 707      0700 2      RETURN
: 708      0701 2
: 709      0702 1      END;
```

				0C	BB	00000	CONVS\$INIT_BUCKET::		
							PUSHR	#M<R2,R3>	0528
							MOVZBL	1(CTX), AREA	0573
							MOVL	CONVS\$AR AREA_BLOCK, R1	0578
							ASHL	#6, AREA, R2	
							PUSHAB	16(R2)[R1]	0580
							PUSHAB	20(R2)[R1]	
							SUBL3	@(SP)+, @(SP)+, R3	
							CMPZV	#0, #8, 3(R2)[R1], R3	0579
							BLEQ	1\$	
							PUSHL	AREA	0585
							BSBW	CONVS\$EXTEND_AREA	
							ADDL2	#4, SP	
							MOVL	CONVS\$AR AREA_BLOCK, R0	0595
							PUSHAB	24(R2)[R0]	
							MOVL	@(SP)+, VBN	
							MOVL	VBN, 8(CTX)	0597
							CML	VBN, #65536	0601
							BGEQ	2\$	
							BICB2	#96, (CTX)	0603
							BRB	4\$	
							CML	VBN, #1048576	0605
							BGEQ	3\$	
							INSV	#1, #5, #2, (CTX)	0607
							BRB	4\$	
							INSV	#2, #5, #2, (CTX)	0609
							MOVZBL	3(R2)[R0], R1	0623
							PUSHAB	20(R2)[R0]	
							ADDL2	R1, @(SP)+	
							PUSHAB	24(R2)[R0]	0627
							ADDL2	R1, @(SP)+	
							PUSHAB	50(R2)[R0]	0631
							ADDL2	R1, @(SP)+	
							CLRB	13(BUCKET)	0641
							MOVW	#14, 4(BUCKET)	0645
							MOVW	8(CTX), 2(BUCKET)	0649
							BLBC	CONVS\$GB_PROL_V3, 6\$	0653
							TSTB	2(CTX)	0659
							BEQL	5\$	
							MOVAB	-92(R10), CTX M1	0668
							EXTZV	#5, #2, (CTX M1), R1	0670
							INSV	R1, #3, #2, T3(BUCKET)	
							MOVZWL	CONVS\$GW VBN FS_PTR, R0	0674
							PUSHAB	(R0)[BUCKET]	
							SUBW3	#1, R0, @(SP)+	
							MOVW	#1, 6(BUCKET)	0680
							BRB	7\$	

CONV\$FSTIO
V04-000

VAX-11 CONVERT
INIT_BUCKET

K 10
15-Sep-1984 23:54:11
14-Sep-1984 12:13:57

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

Page 20
(8)

06	A9		01	90	000B1	6\$:	MOVB	#1, 6(BUCKET)
2A	AA	28	AA	3C	000B5	7\$:	MOVZWL	40(CTX), 42(CTX)
			0C	BA	000BA		POPR	#^M<R2,R3>
				05	000BC		RSB	

: 0688
: 0694
: 0702
:

; Routine Size: 189 bytes, Routine Base: _CONV\$FAST_S + 015D


```
711 0703 1 %SBTTL 'EXTEND_AREA'
712 0704 1 GLOBAL ROUTINE "CONV$$EXTEND_AREA ( AREA ) : CL$EXTEND_AREA NOVALUE =
713 0705 1 ++
714 0706 1
715 0707 1 Functional Description:
716 0708 1
717 0709 1 Extends the disk allocation of a specified area
718 0710 1
719 0711 1 Calling Sequence:
720 0712 1
721 0713 1 EXTEND_AREA ( .area )
722 0714 1
723 0715 1 Input Parameters:
724 0716 1
725 0717 1 AREA - Area to be extended
726 0718 1
727 0719 1 Implicit Inputs:
728 0720 1 none
729 0721 1
730 0722 1 Output Parameters:
731 0723 1 none
732 0724 1
733 0725 1 Implicit Outputs:
734 0726 1 none
735 0727 1
736 0728 1 Routine Value:
737 0729 1 none
738 0730 1
739 0731 1 Routines Called:
740 0732 1
741 0733 1 $EXTEND
742 0734 1 CONV$$RMS_ERROR - By RMS as an AST
743 0735 1
744 0736 1 Side Effects:
745 0737 1 none
746 0738 1
747 0739 1 --
748 0740 1
749 0741 2 BEGIN
750 0742 2
751 0743 2 ! Set the allocation to the largest size. (To avoid bad parameters)
752 0744 2
753 0745 2 CONV$AB_OUT_FAB [ FAB$L_ALQ ] =
754 0746 2 MAX( .CONV$AR_AREA_BLOCK [ .AREA,AREA$W_DEQ ],
755 0747 2 .CONV$AR_AREA_BLOCK [ .AREA,AREA$B_ARBKTSZ ] );
756 0748 2
757 0749 2 ! Wait on the rab in case we have a async operation going on
758 0750 2
759 0751 2 $WAIT( RAB=CONV$AB_OUT_RAB );
760 0752 2
761 0753 2 ! Stuff the error if we get one
762 0754 2
763 0755 2 CONV$AB_OUT_FAB [ FAB$L_CTX ] = CONV$_EXTN_ERR;
764 0756 2
765 0757 2 ! Do the extend
766 0758 2
767 0759 2 $EXTEND ( FAB=CONV$AB_OUT_FAB,ERR=CONV$$RMS_ERROR );
```

```
: 768      0760 2
: 769      0761 2
: 770      0762 2
: 771      0763 2
: 772      0764 2
: 773      0765 2
: 774      0766 2
: 775      0767 2
: 776      0768 2
: 777      0769 2
: 778      0770 2
: 779      0771 1

! Reset some pointers in the prologue
CONVSAR_AREA_BLOCK [ .AREA,AREASL_CVBN ] = .CONVSGL_EOF_VBN;
CONVSAR_AREA_BLOCK [ .AREA,AREASL_NXTVBN ] = .CONVSGL_EOF_VBN;
CONVSAR_AREA_BLOCK [ .AREA,AREASL_CNBLK ] = .CONVSAB_OUT_FAB [ FABSL_ALQ ];
CONVSAR_AREA_BLOCK [ .AREA,AREASL_USED ] = 0;
CONVSGL_EOF_VBN = .CONVSGL_EOF_VBN + .CONVSAB_OUT_FAB [ FABSL_ALQ ];

RETURN

END;
```

.EXTRN SYS\$EXTEND

```
52 DD 00000 CONVS$EXTEND AREA::
                                PUSH R2
                                ASHL #6, AREA, R2
                                ADDL3 CONVSAR_AREA_BLOCK, R2, R1
                                ADDL3 CONVSAR_AREA_BLOCK, R2, R0
                                MOVZWL 36(R1), R1
                                CMPZV #0, #3, 3(R0), R1
                                BLEQ 1$
                                MOVZBL 3(R0), R1
                                MOVL R1, CONVSAB_OUT_FAB+16
                                PUSHAB CONVSAB_OUT_FAB
                                CALLS #1, SYS$WAIT
                                MOVL #CONVS$EXTN_ERR, CONVSAB_OUT_FAB+24
                                PUSHAB CONVS$RMS_ERROR
                                PUSHAB CONVSAB_OUT_FAB
                                CALLS #2, SYS$EXTEND
                                MOVL CONVSAR_AREA_BLOCK, R0
                                PUSHAB 12(R2)[R0]
                                MOVL CONVSGL_EOF_VBN, @ (SP)+
                                PUSHAB 24(R2)[R0]
                                MOVL CONVSGL_EOF_VBN, @ (SP)+
                                PUSHAB 16(R2)[R0]
                                MOVL CONVSAB_OUT_FAB+16, @ (SP)+
                                PUSHAB 20(R2)[R0]
                                CLRL @ (SP)+
                                ADDL2 CONVSAB_OUT_FAB+16, CONVSGL_EOF_VBN
                                POPR #*M<R2>
                                RSB

                                : 0704
                                : 0746
                                : 0747
                                : 0746
                                : 0751
                                : 0755
                                : 0759
                                : 0763
                                : 0764
                                : 0765
                                : 0766
                                : 0767
                                : 0771
                                :
```

; Routine Size: 123 bytes, Routine Base: _CONVSFAST_S + 021A


```

: 781 0772 1 %SBTTL 'CONVERT_VBN_ID'
: 782 0773 1 GLOBAL ROUTINE CONV$$CONVERT_VBN_ID : CL$CONVERT_VBN_ID NOVALUE =
: 783 0774 1 ++
: 784 0775 1
: 785 0776 1 Functional Description:
: 786 0777 1
: 787 0778 1 Converts the rfa created by the sort of the output file
: 788 0779 1 into a VBN and ID to be used as an alternate index pointer
: 789 0780 1
: 790 0781 1 Calling Sequence:
: 791 0782 1
: 792 0783 1 CONV$$CONVERT_VBN_ID()
: 793 0784 1
: 794 0785 1 Input Parameters:
: 795 0786 1 none
: 796 0787 1
: 797 0788 1 Implicit Inputs:
: 798 0789 1 none
: 799 0790 1
: 800 0791 1 Output Parameters:
: 801 0792 1 none
: 802 0793 1
: 803 0794 1 Implicit Outputs:
: 804 0795 1
: 805 0796 1 SORT_VBN - R6 VBN of the primary data record for this key
: 806 0797 1 SORT_ID - R7 ID of the primary data record for this key
: 807 0798 1
: 808 0799 1 Routine Value:
: 809 0800 1 none
: 810 0801 1
: 811 0802 1 Routines Called:
: 812 0803 1 none
: 813 0804 1
: 814 0805 1 Side Effects:
: 815 0806 1 none
: 816 0807 1
: 817 0808 1 --
: 818 0809 1
: 819 0810 2 BEGIN
: 820 0811 2
: 821 0812 2 EXTERNAL REGISTER
: 822 0813 2 SORT_VBN,
: 823 0814 2 SORT_ID;
: 824 0815 2
: 825 0816 2 ! Get the VBN an offset returned by SORT by RFA
: 826 0817 2 !
: 827 0818 2 LOCAL SORT_RFA : REF BLOCK [ ,BYTE ];
: 828 0819 2
: 829 0820 2 SORT_RFA = .CONV$GL_RFA_BUFFER;
: 830 0821 2
: 831 0822 2 SORT_VBN = .SORT_RFA [ 0,0,32,0 ];
: 832 0823 2 SORT_ID = .SORT_RFA [ 4,0,16,0 ];
: 833 0824 2
: 834 0825 2 RETURN
: 835 0826 2
: 836 0827 1 END;
```

```
50      0000G CF DO 00000 CONV$$CONVERT_VBN_ID::
56      60 DO 00005      MOVL CONV$GL_RFA_BUFFER, SORT_RFA      : 0820
57      04 A0 3C 00008      MOVL (SORT_RFA), SORT_VBN      : 0822
      05 0000C      MOVZWL 4(SORT_RFA), SORT_ID      : 0823
      RSB      : 0827
```

; Routine Size: 13 bytes, Routine Base: _CONVF\$FAST_S + 0295

```
; 837      0828 1
; 838      0829 0 END      ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
_CONVF\$FAST_D	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_CONVF\$FAST_S	674	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	40	0	1000	00:01.9
_CONVF\$DUA28:[CONV.SRC]CONVERT.L32;1	165	31	18	17	00:00.2

COMMAND QUALIFIERS

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

```
; Size:      674 code + 4 data bytes
; Run Time:   00:18.4
; Elapsed Time: 01:08.8
; Lines/CPU Min: 2710
; Lexemes/CPU-Min: 19556
; Memory Used: 136 pages
; Compilation Complete
```


0065 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY