


```

LL                      IIIII
LL                      IIIIII
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LLLLLLLLLLLL           IIIIII
LLLLLLLLLLLL           IIIIII

SSSSSSSS              SS
SSSSSSSS              SS
                        SS
                        SS
                        SS
                        SSSSSS
                        SSSSSS
                                SS
                                SS
                                SS
                                SS
SSSSSSSS              SSSSSSSS
SSSSSSSS              SSSSSSSS

```



```
0001 0 %TITLE 'VAX-11 CONVERT'
0002 0 MODULE CONV$COMIO ( 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: Common Convert utilities I/O routines
36 0035 1
37 0036 1 Contents:
38 0037 1 READ PROLOGUE
39 0038 1 WRITE PROLOGUE
40 0039 1 SET_KEY_DESC
41 0040 1 GET_NEXT_KEY
42 0041 1 WRITE_KEY_DESC
43 0042 1 WRITE_AREA_DESC
44 0043 1
45 0044 1 Environment:
46 0045 1
47 0046 1 VAX/VMS Operating System
48 0047 1
49 0048 1 --
50 0049 1
51 0050 1
52 0051 1 Author: Keith B Thompson Creation date: March-1982
53 0052 1
54 0053 1
55 0054 1 Modified by:
56 0055 1
57 0056 1 V03-002 KBT0479 Keith B. Thompson 29-Jan-1983
58 0057 1 Make key_desc_buf and key_desc_vbn global
59 0058 1
60 0059 1 V03-001 KBT0391 Keith B. Thompson 28-Oct-1982
61 0060 1 Make things work
62 0061 1
63 0062 1 !****
```



```
65 0063 1
66 0064 1 PSECT
67 0065 1      OWN      = _CONVSOWN      (PIC),
68 0066 1      GLOBAL   = _CONV$GLOBAL  (PIC),
69 0067 1      PLIT     = _CONV$PLIT    (SHARE,PIC),
70 0068 1      CODE     = _CONV$CODE    (SHARE,PIC);
71 0069 1
72 0070 1 LIBRARY 'SYSSLIBRARY:LIB.L32';
73 0071 1 LIBRARY 'SRCS:CONVERT';
74 0072 1
75 0073 1 DEFINE_ERROR_CODES;
76 0074 1
77 0075 1 LINKAGE
78 0076 1      CL$READ_BLOCK = JSB ( REGISTER = 2, REGISTER = 3 );
79 0077 1      CL$WRITE_BLOCK = JSB ( REGISTER = 2, REGISTER = 3 );
80 0078 1      CL$CHECKSUM   = JSB ( REGISTER = 2 );
81 0079 1
82 0080 1 EXTERNAL ROUTINE
83 0081 1      CONV$$GET_VM      : CL$GET VM,
84 0082 1      CONV$$RMS_ERROR   : NOVALUE,
85 0083 1      CONV$$RMS_OPEN_ERROR : NOVALUE,
86 0084 1      CONV$$RMS_READ_ERROR : NOVALUE;
87 0085 1
88 0086 1 FORWARD ROUTINE
89 0087 1      CONV$$WRITE_AREA_DESC : CL$WRITE AREA_DESC NOVALUE,
90 0088 1      CONV$$GET_NEXT_KEY     : CL$GET NEXT_KEY,
91 0089 1      READ_BLOCK           : CL$READ_BLOCK NOVALUE,
92 0090 1      WRITE_BLOCK          : CL$WRITE_BLOCK NOVALUE,
93 0091 1      CHECKSUM             : CL$CHECKSUM;
94 0092 1
95 0093 1 EXTERNAL
96 0094 1      CONV$AB_FLAGS          : BLOCK [ ,BYTE ],
97 0095 1      CONV$AB_OUT_FAB        : $FAB_DECL,
98 0096 1      CONV$AB_OUT_RAB        : $RAB_DECL,
99 0097 1      CONV$AB_OUT_NAM        : $NAM_DECL,
100 0098 1      CONV$AB_OUT_XABSUM     : $XABSUM_DECL,
101 0099 1      CONV$GB_PROL_V1        : BYTE,
102 0100 1      CONV$GB_PROL_V2        : BYTE,
103 0101 1      CONV$GB_PROL_V3        : BYTE,
104 0102 1      CONV$AR_PROLOGUE     : REF BLOCK [ ,BYTE ],
105 0103 1      CONV$AR_AREA_BLOCK    : REF BLOCKVECTOR [ ,AREASC_BLN,BYTE ];
106 0104 1
107 0105 1 GLOBAL
108 0106 1      CONV$GL_KEY_DESC_BUF,
109 0107 1      CONV$GL_KEY_DESC_VBN;
110 0108 1
111 0109 1 OWN
112 0110 1      AREA_BLOCKS;
113 0111 1
```

```
115 0112 1 %SBTTL 'READ_PROLOGUE'
116 0113 1 GLOBAL ROUTINE CONV$$READ_PROLOGUE : CL$READ_PROLOGUE NOVALUE =
117 0114 1 ++
118 0115 1
119 0116 1 Functional Description:
120 0117 1
121 0118 1 Reads the prologue blocks of the output file. The first block (VBN=1)
122 0119 1 is in the buffer pointed to by conv$ar_prologue. The area descriptors
123 0120 1 are read into the buffer pointed to by conv$ar_area_block. If there
124 0121 1 are more than one key descriptor an extra block is allocated and it
125 0122 1 is pointed to by key_desc_buf.
126 0123 1
127 0124 1 Calling Sequence:
128 0125 1
129 0126 1 conv$$read_prologue()
130 0127 1
131 0128 1 Input Parameters:
132 0129 1 none
133 0130 1
134 0131 1 Implicit Inputs:
135 0132 1 none
136 0133 1
137 0134 1 Output Parameters:
138 0135 1 none
139 0136 1
140 0137 1 Implicit Outputs:
141 0138 1 none
142 0139 1
143 0140 1 Routine Value:
144 0141 1 none
145 0142 1
146 0143 1 Routines Called:
147 0144 1
148 0145 1 CONV$$GET_VM
149 0146 1
150 0147 1 Side Effects:
151 0148 1 none
152 0149 1
153 0150 1 --
154 0151 1
155 0152 2 BEGIN
156 0153 2
157 0154 2 LOCAL
158 0155 2 TOTAL_BLOCKS;
159 0156 2
160 0157 2 The buffer is allocated thus:
161 0158 2
162 0159 2
163 0160 2 conv$ar_prologue :
164 0161 2 512 Bytes
165 0162 2
166 0163 2
167 0164 2 conv$gl_key_desc_buf :
168 0165 2 512 Bytes
169 0166 2
170 0167 2
171 0168 2 conv$ar_area_block :
```



```
172 0169 2 |
173 0170 2 |
174 0171 2 |
175 0172 2 |
176 0173 2 |
177 0174 2 |
178 0175 2 |
179 0176 2 |
180 0177 2 |
181 0178 2 |
182 0179 2 |
183 0180 2 |
184 0181 2 |
185 0182 2 |
186 0183 2 |
187 0184 2 |
188 0185 2 |
189 0186 2 |
190 0187 2 |
191 0188 2 |
192 0189 2 |
193 0190 2 |
194 0191 2 |
195 0192 2 |
196 0193 2 |
197 0194 2 |
198 0195 2 |
199 0196 2 |
200 0197 2 |
201 0198 2 |
202 0199 2 |
203 0200 2 |
204 0201 2 |
205 0202 2 |
206 0203 2 |
207 0204 2 |
208 0205 2 |
209 0206 2 |
210 0207 2 |
211 0208 2 |
212 0209 2 |
213 0210 2 |
214 0211 2 |
215 0212 2 |
216 0213 2 |
217 0214 2 |
218 0215 2 |
219 0216 2 |
220 0217 2 |
221 0218 2 |
222 0219 1 |
```

512*No. of
Area Blocks

Figure out the number of blocks for the prologue area desc.
AREA_BLOCKS = ((.CONV\$AB_OUT_XABSUM [XAB\$B_NOA] - 1) / 8) + 1;
! The total blocks is area blocks + prologue block + key desc buffer
TOTAL_BLOCKS = .AREA_BLOCKS + 1 + 1;
! Get the address space.
CONV\$AR_PROLOGUE = CONV\$\$GET_VM(.TOTAL_BLOCKS * BLOCK_SIZE);
! The key block points just after the prologue block
CONV\$GL_KEY_DESC_BUF = .CONV\$AR_PROLOGUE + BLOCK_SIZE;
! The area descriptors is after everything
CONV\$AR_AREA_BLOCK = .CONV\$GL_KEY_DESC_BUF + BLOCK_SIZE;
! Read in the prologue block
READ_BLOCK(.CONV\$AR_PROLOGUE, 1);
! Read each of the area blocks
INCR I FROM 0 TO .AREA_BLOCKS - 1
DO
 READ_BLOCK(.CONV\$AR_AREA_BLOCK + (.I * BLOCK_SIZE),
 .CONV\$AR_PROLOGUE [PLG\$B_AVBN] + .I);
! Set the proper prologue version flag
SELECTONE .CONV\$AR_PROLOGUE [PLG\$W_VER_NO] OF
SET
 [PLG\$C_VER_NO] : CONV\$GB_PROL_V1 = SET;
 [PLG\$C_VER_IDX] : CONV\$GB_PROL_V2 = SET;
 [PLG\$C_VER_3] : CONV\$GB_PROL_V3 = SET;
 [OTHERWISE] : SIGNAL_STOP(CONV\$PLV);
TES;
RETURN
END;

.TITLE CONV\$COMIO VAX-11 CONVERT
.IDENT \V04-000\
.PSECT _CONV\$GLOBAL,NOEXE, PIC,2

00000 CONV\$GL_KEY_DESC_BUF::

.BLRB 4
00004 CONV\$GL_KEY_DESC_VBN::

.PSECT _CONVSOWN,NOEXE, PIC,2

00000 AREA_BLOCKS:

.BLKB 4

.EXTRN CONVERTS 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_VM, CONV\$SRMS_ERROR
.EXTRN CONV\$SRMS_OPEN_ERROR
.EXTRN CONV\$SRMS_READ_ERROR
.EXTRN CONV\$AB_FLAGS, CONV\$AB_OUT_FAB
.EXTRN CONV\$AB_OUT_RAB
.EXTRN CONV\$AB_OUT_NAM
.EXTRN CONV\$AB_OUT_XABSUM
.EXTRN CONV\$GB_PROL_V1
.EXTRN CONV\$GB_PROL_V2
.EXTRN CONV\$GB_PROL_V3
.EXTRN CONV\$AR_PROLOGUE
.EXTRN CONV\$AR_AREA_BLOCK

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

3C BB 00000 CONV\$\$READ PROLOGUE::

50 0000G CF 9A 00002 PUSHR #*M<R2,R3,R4,R5>
50 D7 00007 MOVZBL CONV\$AB_OUT_XABSUM+8, R0
50 08 C6 00009 DECL R0
0000' CF 01 A0 9E 0000C DIVL2 #8, R0
MOVAB 1(R0), AREA_BLOCKS

: 0113
: 0178
: :
: :

50	0000'	CF	02	C1	00012	ADDL3	#2, AREA_BLOCKS, TOTAL_BLOCKS	0182
7E		50	09	78	00018	ASHL	#9, TOTAL_BLOCKS, -(SP)	0186
			0000G	30	0001C	BSBW	CONV\$GET_VM	
		5E	04	C0	0001F	ADDL2	#4, SP	
0000'	CF	0000G	50	D0	00022	MOVL	R0, CONV\$AR_PROLOGUE	
		0000G	8F	C1	00027	ADDL3	#512, CONV\$AR_PROLOGUE, -	0190
							CONV\$GL_KEY_DESC_BUF	
0000G	CF	0000'	8F	C1	00033	ADDL3	#512, CONV\$GL_KEY_DESC_BUF, -	0194
							CONV\$AR_AREA_BLOCK	
		53	01	D0	0003F	MOVL	#1, R3	0198
		52	0000G	CF	00042	MOVL	CONV\$AR_PROLOGUE, R2	
			0000V	30	00047	BSBW	READ_BLOCK	
		55	0000'	CF	0004A	MOVL	AREA_BLOCKS, R5	0202
		54		01	CE	MNEGL	#1, I	
				1D	11	BRB	2\$	
		50	0000G	CF	00054	MOVL	CONV\$AR_PROLOGUE, R0	0205
		50	66	A0	9E	MOVAB	102(R0), R0	
		50		60	9A	MOVZBL	(R0), R0	
53		50		54	C1	ADDL3	R0, R3	
50		54		09	78	ASHL	#9, I, R0	0204
52		50	0000G	CF	00068	ADDL3	CONV\$AR_AREA_BLOCK, R0, R2	
			0000V	30	0006E	BSBW	READ_BLOCK	
DF		54		55	F2	AOBLSS	R5, I, 1\$	
		50	0000G	CF	00075	MOVL	CONV\$AR_PROLOGUE, R0	0209
		50	74	A0	9E	MOVAB	116(R0), R0	
		50		60	3C	MOVZWL	(R0), R0	
		01		50	B1	CMPW	R0, #1	0211
				07	12	BNEQ	3\$	
		0000G	CF	01	90	MOVB	#1, CONV\$GB_PROL_V1	
				25	11	BRB	6\$	
		02		50	B1	CMPW	R0, #2	0212
				07	12	BNEQ	4\$	
		0000G	CF	01	90	MOVB	#1, CONV\$GB_PROL_V2	
				19	11	BRB	6\$	
		03		50	B1	CMPW	R0, #3	0213
				07	12	BNEQ	5\$	
		0000G	CF	01	90	MOVB	#1, CONV\$GB_PROL_V3	
				0D	11	BRB	6\$	
			00000000G	8F	DD	PUSHL	#CONV\$ PLV	0214
		00000000G	00	01	FB	CALLS	#1, LIB\$STOP	
				3C	BA	POPR	#^M<R2,R3,R4,R5>	0219
				05	000B4	RSB		

; Routine Size: 181 bytes, Routine Base: _CONV\$CODE + 0000

```
224 0220 1 %SBTTL 'WRITE_PROLOGUE'
225 0221 1 GLOBAL ROUTINE CONV$$WRITE_PROLOGUE : NOVALUE =
226 0222 1 ++
227 0223 1
228 0224 1 Functional Description:
229 0225 1
230 0226 1 Writes the prologue area blocks back to the output file
231 0227 1
232 0228 1 Calling Sequence:
233 0229 1
234 0230 1 CONV$$WRITE_PROLOGUE()
235 0231 1
236 0232 1 Input Parameters:
237 0233 1 none
238 0234 1
239 0235 1 Implicit Inputs:
240 0236 1 none
241 0237 1
242 0238 1 Output Parameters:
243 0239 1 none
244 0240 1
245 0241 1 Implicit Outputs:
246 0242 1 none
247 0243 1
248 0244 1 Routine Value:
249 0245 1 none
250 0246 1
251 0247 1 Routines Called:
252 0248 1
253 0249 1 WRITE_BLOCK
254 0250 1
255 0251 1 Side Effects:
256 0252 1 none
257 0253 1
258 0254 1 --
259 0255 1
260 0256 2 BEGIN
261 0257 2
262 0258 2 ! Write each of the area blocks
263 0259 2 !
264 0260 2 INCR I FROM 0 TO .AREA_BLOCKS - 1
265 0261 2 DO
266 0262 2 WRITE_BLOCK( .CONV$AR_AREA_BLOCK + ( .I * BLOCK_SIZE ),
267 0263 2 .CONV$AR_PROLOGUE [ PLG$B_AVBN ] + .I );
268 0264 2
269 0265 2
270 0266 2 RETURN
271 0267 2
272 0268 1 END;
```

```
OFFC 00000
55 0000' CF DO 00002
```

```
.ENTRY CONV$$WRITE_PROLOGUE, Save R2,R3,R4,R5,R6,- ; 0221
R7,R8,R9,R10,R11 ;
MOVL AREA_BLOCKS, R5 ; 0260
```


CONV\$COMIO
V04-000

VAX-11 CONVERT
WRITE_PROLOGUE

M 16
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

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

Page 9
(5)

	54		01	CE	00007		MNEGL	#1, 1		
			1D	11	0000A		BRB	2\$		
	50	0000G	CF	D0	0000C	1\$:	MOVL	CONVSAR_PROLOGUE, R0		0263
	50	66	A0	9E	00011		MOVAB	102(R0), R0		
	50		60	9A	00015		MOVZBL	(R0), R0		
53	50		54	C1	00018		ADDL3	1, R0, R3		
50	54		09	78	0001C		ASHL	#9, 1, R0		0262
52	50	0000G	CF	C1	00020		ADDL3	CONVSAR_AREA_BLOCK, R0, R2		
			0000V	30	00026		BSBW	WRITE_BLOCK		
DF	54		55	F2	00029	2\$:	A0BLSS	R5, 1, 1\$		
			04	0002D			RET			0268

; Routine Size: 46 bytes, Routine Base: _CONV\$CODE + 00B5


```
274 0269 1 %SBTTL 'SET KEY DESC'
275 0270 1 GLOBAL ROUTINE CONV$$SET_KEY_DESC ( KEY ) : CL$SET_KEY_DESC =
276 0271 1 **
277 0272 1
278 0273 1 Functional Description:
279 0274 1
280 0275 1     Sets the key descriptor from the output files
281 0276 1     prologue to the requested key of reference.
282 0277 1     This routine WILL reread the key descriptor from
283 0278 1     the file.
284 0279 1
285 0280 1 Calling Sequence:
286 0281 1
287 0282 1     CONV$$SET_KEY_DESC( key )
288 0283 1
289 0284 1 Input Parameters:
290 0285 1
291 0286 1     key      - Key of refrence to get
292 0287 1
293 0288 1 Implicit Inputs:
294 0289 1
295 0290 1     CONV$GL_KEY_DESC_BUF
296 0291 1
297 0292 1 Output Parameters:
298 0293 1     none
299 0294 1
300 0295 1 Implicit Outputs:
301 0296 1
302 0297 1     KEY_DESC
303 0298 1     CONV$GL_KEY_DESC_VBN
304 0299 1
305 0300 1 Routine Value:
306 0301 1
307 0302 1     CONV$_SUCCESS or CONV$_NOKEY (from get_next_key)
308 0303 1
309 0304 1 Routines Called:
310 0305 1
311 0306 1     CONV$_GET_NEXT_KEY
312 0307 1
313 0308 1 Side Effect:
314 0309 1     none
315 0310 1
316 0311 1 --
317 0312 1
318 0313 2 BEGIN
319 0314 2
320 0315 2 DEFINE_KEY_DESC;
321 0316 2
322 0317 2 LOCAL      STATUS;
323 0318 2
324 0319 2 STATUS = CONV$_SUCCESS;
325 0320 2
326 0321 2 ! Reset to the primary key then search from there
327 0322 2 !
328 0323 2 KEY_DESC = .CONV$GL_KEY_DESC_BUF;
329 0324 2
330 0325 2 CONV$GL_KEY_DESC_VBN = 1;
```



```
... 331 0326 2
... 332 0327 2
... 333 0328 2
... 334 0329 2
... 335 0330 2
... 336 0331 2
... 337 0332 2
... 338 0333 2
... 339 0334 2
... 340 0335 2
... 341 0336 2
... 342 0337 2
... 343 0338 2
... 344 0339 2
... 345 0340 2
... 346 0341 2
... 347 0342 1

! Read the first key
! READ_BLOCK( .KEY_DESC, .CONV$GL_KEY_DESC_VBN );
! Loop until you find the correct key
! WHILE .STATUS AND ( .KEY NEQU .KEY_DESC [ KEY$B_KEYREF ] )
DO
    ! If there are no keys then what a bummer
    ! STATUS = CONV$$GET_NEXT_KEY();
RETURN .STATUS
END;
```

			1C	BB	00000	CONV\$\$SET_KEY_DESC::			
						PUSHR	#^M<R2,R3,R4>	: 0270	
		54	01	DO	00002	MOVL	#1, STATUS	: 0319	
		5B	CF	DO	00005	MOVL	CONV\$GL_KEY_DESC_BUF, KEY_DESC	: 0323	
	0000'	CF	01	DO	0000A	MOVL	#1, CONV\$GL_KEY_DESC_VBN	: 0325	
		53	CF	DO	0000F	MOVL	CONV\$GL_KEY_DESC_VBN, R3	: 0329	
		52	5B	DO	00014	MOVL	KEY_DESC, R2		
			0000V	30	00017	BSBW	READ_BLOCK		
		11	54	E9	0001A	1\$:	BLBC	STATUS, 2\$	: 0333
10	AE		08	00	ED	0001D	CMPZV	#0, #8, 21(KEY_DESC), KEY	
			08	13	00024	BEQL	2\$		
			0000V	30	00026	BSBW	CONV\$\$GET_NEXT_KEY	: 0338	
		54	50	DO	00029	MOVL	R0, STATUS		
			EC	11	0002C	BRB	1\$		
		50	54	DO	0002E	2\$:	MOVL	STATUS, R0	: 0340
			1C	BA	00031	POPR	#^M<R2,R3,R4>	: 0342	
				05	00033	RSB			

; Routine Size: 52 bytes, Routine Base: _CONV\$CODE + 00E3


```

349 0343 1 %SBTTL 'GET NEXT KEY'
350 0344 1 GLOBAL ROUTINE CONV$$GET_NEXT_KEY : CL$GET_NEXT_KEY =
351 0345 1 ++
352 0346 1
353 0347 1 Functional Description:
354 0348 1
355 0349 1     Sets the key descriptor from the output files
356 0350 1     prologue to the next key of reference if any
357 0351 1
358 0352 1 Calling Sequence:
359 0353 1
360 0354 1     CONV$$GET_NEXT_KEY()
361 0355 1
362 0356 1 Input Parameters:
363 0357 1     none
364 0358 1
365 0359 1 Implicit Inputs:
366 0360 1
367 0361 1     KEY_DESC
368 0362 1
369 0363 1 Output Parameters:
370 0364 1     none
371 0365 1
372 0366 1 Implicit Outputs:
373 0367 1
374 0368 1     KEY_DESC
375 0369 1
376 0370 1 Routine Value:
377 0371 1
378 0372 1     CONV$_SUCCESS or CONV$_NOKEY
379 0373 1
380 0374 1 Routines Called:
381 0375 1
382 0376 1     READ_BLOCK
383 0377 1
384 0378 1 Side Effects:
385 0379 1
386 0380 1     Could read a new key descriptor into memory
387 0381 1
388 0382 1 --
389 0383 1
390 0384 2 BEGIN
391 0385 2
392 0386 2 DEFINE_KEY_DESC:
393 0387 2
394 0388 2 ! If the next key in the chain is not in this block
395 0389 2 ! then get the next block in the chain
396 0390 2
397 0391 2 IF .KEY_DESC [ KEY$_IDXFL ] NEQ 0
398 0392 2 THEN
399 0393 2 BEGIN
400 0394 2
401 0395 2 ! Get the VBN of the next block
402 0396 2
403 0397 2 CONV$GL_KEY_DESC_VBN = .KEY_DESC [ KEY$_IDXFL ];
404 0398 2
405 0399 2 ! Have key block point to the right place in the new block
```



```

: 406      0400      :
: 407      0401      : KEY_DESC = .CONV$GL_KEY_DESC_BUF + .KEY_DESC [ KEY$W_NOFF ];
: 408      0402      :
: 409      0403      : Read the block
: 410      0404      :
: 411      0405      : READ_BLOCK( .CONV$GL_KEY_DESC_BUF, .CONV$GL_KEY_DESC_VBN )
: 412      0406      :
: 413      0407      : END
: 414      0408      ELSE
: 415      0409      :
: 416      0410      : If the offset is 0 then there are no more keys
: 417      0411      :
: 418      0412      : IF .KEY_DESC [ KEY$W_NOFF ] EQL 0
: 419      0413      : THEN
: 420      0414      : RETURN CONV$_NOKEY
: 421      0415      : ELSE
: 422      0416      :
: 423      0417      : Point the key block to the next key descriptor
: 424      0418      :
: 425      0419      : KEY_DESC = .CONV$GL_KEY_DESC_BUF + .KEY_DESC [ KEY$W_NOFF ];
: 426      0420      :
: 427      0421      : RETURN CONV$_SUCCESS
: 428      0422      :
: 429      0423      : END;
```

			OC	BB	00000	CONV\$\$GET_NEXT_KEY::		
			6B	D5	00002	PUSHR	#^M<R2,R3>	: 0344
			18	13	00004	TSTL	(KEY_DESC)	: 0391
0000'	CF		6B	D0	00006	BEQL	1\$	: 0397
	5B	04	AB	3C	0000B	MOVL	(KEY_DESC), CONV\$GL_KEY_DESC_VBN	: 0401
	5B	0000'	CF	C0	0000F	MOVZWL	4(KEY_DESC), KEY_DESC	: 0405
	52	0000'	CF	7D	00014	ADDL2	CONV\$GL_KEY_DESC_BUF, KEY_DESC	: 0412
		0000V	30	00019	MOVQ	CONV\$GL_KEY_DESC_BUF, R2	: 0414	
			17	11	0001C	BSBW	READ_BLOCK	: 0419
		04	AB	B5	0001E	BRB	3\$	: 0421
			09	12	00021	TSTW	4(KEY_DESC)	: 0423
50	00000000G		8F	D0	00023	BNEQ	2\$	
			0C	11	0002A	MOVL	#CONV\$_NOKEY, R0	
5B	04		AB	3C	0002C	BRB	4\$	
5B	0000'		CF	C0	00030	MOVZWL	4(KEY_DESC), KEY_DESC	
50			01	D0	00035	ADDL2	CONV\$GL_KEY_DESC_BUF, KEY_DESC	
			0C	BA	00038	MOVL	#1, R0	
			05	0003A	POPR	#^M<R2,R3>		
					RSB			

; Routine Size: 59 bytes, Routine Base: _CONV\$CODE + 0117


```
431 0424 1 %SBTTL 'WRITE_KEY_DESC'
432 0425 1 GLOBAL ROUTINE CONV$$WRITE_KEY_DESC : CL$WRITE_KEY_DESC NOVALUE =
433 0426 1 ++
434 0427 1
435 0428 1 Functional Description:
436 0429 1
437 0430 1 Writes back to the output file the current key descriptor
438 0431 1
439 0432 1 Calling Sequence:
440 0433 1
441 0434 1 CONV$$WRITE_KEY_DESC()
442 0435 1
443 0436 1 Input Parameters:
444 0437 1 none
445 0438 1
446 0439 1 Implicit Inputs:
447 0440 1
448 0441 1 CONV$GL_KEY_DESC_BUF
449 0442 1 CONV$GL_KEY_DESC_VBN
450 0443 1
451 0444 1 Output Parameters:
452 0445 1 none
453 0446 1
454 0447 1 Implicit Outputs:
455 0448 1 none
456 0449 1
457 0450 1 Routine Value:
458 0451 1 none
459 0452 1
460 0453 1 Routines Called:
461 0454 1
462 0455 1 WRITE_BLOCK
463 0456 1
464 0457 1 Side Effects:
465 0458 1 none
466 0459 1
467 0460 1 --
468 0461 1
469 0462 2 BEGIN
470 0463 2
471 0464 2 WRITE_BLOCK( .CONV$GL_KEY_DESC_BUF,.CONV$GL_KEY_DESC_VBN );
472 0465 2
473 0466 2 RETURN
474 0467 2
475 0468 1 END;
```

```
OC BB 00000 CONV$$WRITE_KEY_DESC::
52 0000 CF 7D 00002 PUSRR #^M<R2,R3>
0000V 30 00007 MOVQ CONV$GL_KEY_DESC_BUF, R2
OC BA 0000A BSBW WRITE_BLOCK
05 0000C POPR #^M<R2,R3>
RSB
```

```
: 0425
: 0464
: 0468
:
```


CONV\$COMIO
V04-000

VAX-11 CONVERT
WRITE_KEY_DESC

6 1
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

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

Page 15
(8)

; Routine Size: 13 bytes, Routine Base: _CONV\$CODE + 0152


```
477 0469 1 %SBTTL 'WRITE_AREA_DESC'
478 0470 1 GLOBAL ROUTINE CONV$$WRITE_AREA_DESC ( AREA ) : CL$WRITE_AREA_DESC NOVALUE =
479 0471 1 ++
480 0472 1
481 0473 1 Functional Description:
482 0474 1
483 0475 1 Writes back to the output file the current key descriptor
484 0476 1
485 0477 1 Calling Sequence:
486 0478 1
487 0479 1 CONV$$WRITE_AREA_DESC( AREA )
488 0480 1
489 0481 1 Input Parameters:
490 0482 1
491 0483 1 AREA - Area number to write
492 0484 1
493 0485 1 Implicit Inputs:
494 0486 1
495 0487 1 CONV$AR_AREA_BLOCK
496 0488 1
497 0489 1 Output Parameters:
498 0490 1 none
499 0491 1
500 0492 1 Implicit Outputs:
501 0493 1 none
502 0494 1
503 0495 1 Routine Value:
504 0496 1 none
505 0497 1
506 0498 1 Routines Called:
507 0499 1
508 0500 1 WRITE_BLOCK
509 0501 1
510 0502 1 Side Effects:
511 0503 1 none
512 0504 1
513 0505 1 --
514 0506 1
515 0507 2 BEGIN
516 0508 2
517 0509 2 LOCAL
518 0510 2 VBN,
519 0511 2 BUFFER;
520 0512 2
521 0513 2 ! Determine what block the area descriptor is in
522 0514 2 !
523 0515 2 VBN = .CONV$AR_PROLOGUE [ PLG$B_AVBN ] + ( ( .AREA - 1 ) / 8 );
524 0516 2
525 0517 2 ! Where in the buffer is the area descriptor
526 0518 2 !
527 0519 2 BUFFER = .CONV$AR_AREA_BLOCK +
528 0520 2 ( ( .VBN - .CONV$AR_PROLOGUE [ PLG$B_AVBN ] ) * BLOCK_SIZE );
529 0521 2
530 0522 2 WRITE_BLOCK( .BUFFER, .VBN );
531 0523 2
532 0524 2 RETURN
533 0525 2
```


CONV\$COM10
V04-000

VAX-11 CONVERT
WRITE_AREA_DESC

I 1
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

VAX-11 Bliss-32 V4.0-742
[CONV.SRC]CONVCOM10.B32;1

Page 17
(9)

: 534

0526 1 END;

		OC	BB	00000	CONV\$\$WRITE AREA DESC::	
					PUSRR #^M<R2,R3>	: 0470
50	0000G	CF	D0	00002	MOVL CONV\$AR_PROLOGUE, R0	: 0515
		51	D7	00007	DECL R1	:
51		08	C6	00009	DIVL2 #8, R1	:
52	66	A0	9A	0000C	MOVZBL 102(R0), R2	:
51		52	C0	00010	ADDL2 R2, VBN	:
53	66	A0	9A	00013	MOVZBL 102(R0), R3	: 0520
53		53	C3	00017	SUBL3 R3, VBN, R3	:
53		09	78	0001B	ASHL #9, R3, R3	:
53	0000G	CF	C1	0001F	ADDL3 CONV\$AR_AREA_BLOCK, R3, BUFFER	:
53		51	D0	00025	MOVL VBN, R3	: 0522
		0000V	30	00028	BSBW WRITE_BLOCK	:
		OC	BA	0002B	POPR #^M<R2,R3>	: 0526
			05	0002D	RSB	:

; Routine Size: 46 bytes, Routine Base: _CONV\$CODE + 015F


```
536 0527 1 %SBTTL 'READ_BLOCK'
537 0528 1 ROUTINE READ_BLOCK ( BUFFER : REF VECTOR [ ,WORD ],VBN ) : CL$READ_BLOCK NOVALUE =
538 0529 1 ++
539 0530 1
540 0531 1 Functional Description:
541 0532 1
542 0533 1 Reads a block in the output files prologue and checks the
543 0534 1 checksum value for it
544 0535 1
545 0536 1 Calling Sequence:
546 0537 1
547 0538 1 READ_BLOCK( buffer,vbn )
548 0539 1
549 0540 1 Input Parameters:
550 0541 1
551 0542 1 buffer - Buffer to read the block into
552 0543 1 vbn - VBN in the prologue to read
553 0544 1
554 0545 1 Implicit Inputs:
555 0546 1 none
556 0547 1
557 0548 1 Output Parameters:
558 0549 1 none
559 0550 1
560 0551 1 Implicit Outputs:
561 0552 1 none
562 0553 1
563 0554 1 Routine Value:
564 0555 1 none
565 0556 1
566 0557 1 Routines Called:
567 0558 1
568 0559 1 CHECKSUM
569 0560 1
570 0561 1 Side Effects:
571 0562 1 none
572 0563 1
573 0564 1 --
574 0565 1
575 0566 2 BEGIN
576 0567 2
577 0568 2 CONV$AB_OUT_RAB [ RAB$L_BKT ] = .VBN;
578 0569 2 CONV$AB_OUT_RAB [ RAB$L_UBF ] = .BUFFER;
579 0570 2 CONV$AB_OUT_RAB [ RAB$W_USZ ] = BLOCK_SIZE;
580 0571 2
581 0572 2 $READ( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_READ_ERROR );
582 0573 2
583 0574 2 IF .BUFFER [ 255 ] NEQU CHECKSUM( .BUFFER )
584 0575 2 THEN
585 0576 2 BEGIN
586 0577 2
587 0578 2 LOCAL FILE_NAME : DESC_BLK;
588 0579 2
589 0580 2 ! The file is open so there should be a full name around
590 0581 2 !
591 0582 2 FILE_NAME [ DSC$W_LENGTH ] = .CONV$AB_OUT_NAM [ NAM$B_RSL ];
592 0583 2 FILE_NAME [ DSC$A_POINTER ] = .CONV$AB_OUT_NAM [ NAM$C_RSA ];
```


CONV\$COM10
V04-000

VAX-11 CONVERT
READ_BLOCK

K 1
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

VAX-11 Bliss-32 V4.0-742
[CONV.SRC]CONVCOM10.B32;1

Page 19
(10)

```
: 593      0584      3
: 594      0585      3      SIGNAL_STOP( CONV$_READERR,1,FILE_NAME,CONV$_PROERR,1,.VBN )
: 595      0586      3
: 596      0587      3      END;
: 597      0588      3
: 598      0589      3      RETURN
: 599      0590      3
: 600      0591      1      END;
```

```
                                .EXTRN  SYS$READ
                                5E      08  C2 00000 READ_BLOCK:
                                0000G CF 53 D0 00003      SUBL2      #8, SP      : 0528
                                0000G CF 52 D0 00008      MOVL      VBN, CONV$AB_OUT_RAB+56      : 0568
                                0000G CF 0200 8F B0 0000D      MOVL      BUFFER, CONV$AB_OUT_RAB+36      : 0569
                                0000G CF 0000G 9F 00014      MOVW      #512, CONV$AB_OUT_RAB+32      : 0570
                                0000G CF 9F 00018      PUSHAB     CONV$SRMS_READ_ERROR      : 0572
                                00000000G 00 02 FB 0001C      PUSHAB     CONV$AB_OUT_RAB      :
                                0000V 30 00023      CALLS      #2, SYS$READ      :
                                50      01FE  C2      00 00 ED 00026      BSBW      CHECKSUM      : 0574
                                10      27 13 0002D      CMPZV     #0, #16, 510(BUFFER), R0      :
                                04      6E      0000G CF 9B 0002F      BEQL      1$      :
                                AE      0000G CF D0 00034      MOVZBW     CONV$AB_OUT_NAM+3, FILE_NAME      : 0582
                                0000G CF D0 00034      MOVL      CONV$AB_OUT_NAM+4, FILE_NAME+4      : 0583
                                53 DD 0003A      PUSHL     VBN      : 0585
                                01 DD 0003C      PUSHL     #1
                                00000000G 8F DD 0003E      PUSHL     #CONV$ PROERR
                                OC AE 9F 00044      PUSHAB     FILE_NAME
                                01 DD 00047      PUSHL     #1
                                00000000G 8F DD 00049      PUSHL     #CONV$ READERR
                                00 06 FB 0004F      CALLS      #6, LIB$STOP
                                SE 08 C0 00056 1$:      ADDL2     #8, SP
                                05 00059      RSB      : 0591
```

; Routine Size: 90 bytes, Routine Base: _CONV\$CODE + 018D


```
.. 602      0592 1 %SBTTL 'WRITE_BLOCK'
.. 603      0593 1 ROUTINE WRITE_BLOCK ( BUFFER : REF VECTOR[WORD ],VBN ) : CL$WRITE_BLOCK NOVALUE =
.. 604      0594 1 ++
.. 605      0595 1
.. 606      0596 1 Functional Description:
.. 607      0597 1
.. 608      0598 1     Calculates a checksum for a block and writes the block to
.. 609      0599 1     the output files prologue
.. 610      0600 1
.. 611      0601 1 Calling Sequence:
.. 612      0602 1
.. 613      0603 1     WRITE_BLOCK( buffer,vbn )
.. 614      0604 1
.. 615      0605 1 Input Parameters:
.. 616      0606 1
.. 617      0607 1     buffer - Buffer to write the block from
.. 618      0608 1     vbn    - VBN in the prologue to write
.. 619      0609 1
.. 620      0610 1 Implicit Inputs:
.. 621      0611 1     none
.. 622      0612 1
.. 623      0613 1 Output Parameters:
.. 624      0614 1     none
.. 625      0615 1
.. 626      0616 1 Implicit Outputs:
.. 627      0617 1     none
.. 628      0618 1
.. 629      0619 1 Routine Value:
.. 630      0620 1     none
.. 631      0621 1
.. 632      0622 1 Routines Called:
.. 633      0623 1
.. 634      0624 1     CHECKSUM
.. 635      0625 1
.. 636      0626 1 Side Effects:
.. 637      0627 1     none
.. 638      0628 1
.. 639      0629 1 --
.. 640      0630 1
.. 641      0631 2 BEGIN
.. 642      0632 2
.. 643      0633 2     BUFFER [ 255 ] = CHECKSUM ( .BUFFER );
.. 644      0634 2
.. 645      0635 2     CONV$AB_OUT_RAB [ RAB$B_BKT ] = .VBN;
.. 646      0636 2     CONV$AB_OUT_RAB [ RAB$B_RBF ] = .BUFFER;
.. 647      0637 2     CONV$AB_OUT_RAB [ RAB$B_RSZ ] = BLOCK_SIZE;
.. 648      0638 2
.. 649      0639 2     ! It's ok to call rms_read_error it works for writes to
.. 650      0640 2     !
.. 651      0641 2     $WRITE( RAB=CONV$AB_OUT_RAB,ERR=CONV$$RMS_READ_ERROR );
.. 652      0642 2
.. 653      0643 2 RETURN
.. 654      0644 2
.. 655      0645 1 END;
```



```
VAX-11 CONVERT
WRITE_BLOCK
```

M 1
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

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

Page 21
(11)

```
.EXTRN  SYSSWRITE
```

0000V 30 00000 WRITE_BLOCK:

01FE	C2	50	B0	00003
0000G	CF	53	D0	00008
0000G	CF	52	D0	0000D
0000G	CF	0200	8F	B0 00012
		0000G	CF	9F 00019
		0000G	CF	9F 0001D
00000000G	00		02	FB 00021
				05 00028

```

BSBW
MOVW
MOVL
MOVL
MOVW
PUSHAB
PUSHAB
CALLS
RSB
CHECKSUM
R0, 510(BUFFER)
VBN, CONV$AB OUT RAB+56
BUFFER, CONV$AB OUT RAB+40
#512, CONV$AB OUT RAB+34
CONV$$RMS READ ERROR
CONV$AB OUT RAB
#2, SYS$WRITE

```

0633
0635
0636
0637
0641
0645

```
; Routine Size: 41 bytes,    Routine Base: _CONV$CODE + 01E7
```



```

: 657 0646 1 %SBTTL 'CHECKSUM'
: 658 0647 1 ROUTINE CHECKSUM ( BLOCK : REF VECTOR [ ,WORD ] ) : CL$CHECKSUM =
: 659 0648 1 ++
: 660 0649 1
: 661 0650 1 Functional Description:
: 662 0651 1
: 663 0652 1     Calculates a checksum for a block and writes the block to
: 664 0653 1     the output files prologue
: 665 0654 1
: 666 0655 1 Calling Sequence:
: 667 0656 1
: 668 0657 1     CHECKSUM( buffer )
: 669 0658 1
: 670 0659 1 Input Parameters:
: 671 0660 1
: 672 0661 1     buffer - 512 byte buffer to calculate the checksum for
: 673 0662 1
: 674 0663 1 Implicit Inputs:
: 675 0664 1     none
: 676 0665 1
: 677 0666 1 Output Parameters:
: 678 0667 1     none
: 679 0668 1
: 680 0669 1 Implicit Outputs:
: 681 0670 1     none
: 682 0671 1
: 683 0672 1 Routine Value:
: 684 0673 1
: 685 0674 1     R0 - Checksum
: 686 0675 1
: 687 0676 1 Routines Called:
: 688 0677 1     none
: 689 0678 1
: 690 0679 1 Side Effects:
: 691 0680 1     none
: 692 0681 1
: 693 0682 1 --
: 694 0683 1
: 695 0684 2 BEGIN
: 696 0685 2
: 697 0686 2 ! Calculate the checksum for this block
: 698 0687 2 !
: 699 0688 2 LOCAL CHECKSUM : WORD;
: 700 0689 2
: 701 0690 2 CHECKSUM = 0;
: 702 0691 2
: 703 0692 2 INCR J FROM 0 TO 254 BY 1
: 704 0693 2 DO
: 705 0694 2     CHECKSUM = .CHECKSUM + .BLOCK [ .J ];
: 706 0695 2
: 707 0696 2 RETURN .CHECKSUM
: 708 0697 2
: 709 0698 1 END;
```


CONV\$COMIO
V04-000

VAX-11 CONVERT
CHECKSUM

B 2
15-Sep-1984 23:47:13
14-Sep-1984 12:13:48

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

Page 23
(12)

51 84 00000 CHECKSUM:

F4 51 000000FE 50 000002 50 D4 00002 1\$: CLRW CHECKSUM
50 8F F3 00008 ADDW2 J (BLOCK)[J], CHECKSUM
50 51 3C 00010 AOBLEQ #254, J, 1\$
05 00013 MOVZWL CHECKSUM, R0
RSB

: 0690
: 0694
: 0696
: 0698

; Routine Size: 20 bytes, Routine Base: _CONV\$CODE + 0210

: 710 0699 1
: 711 0700 0 END ELUDOM

.EXTRN LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
_CONV\$GLOBAL	8	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_CONV\$OWN	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_CONV\$CODE	548	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	32	0	1000	00:01.8
-\$255\$DUA28:[CONV.SRC]CONVERT.L32;1	165	12	7	17	00:00.2

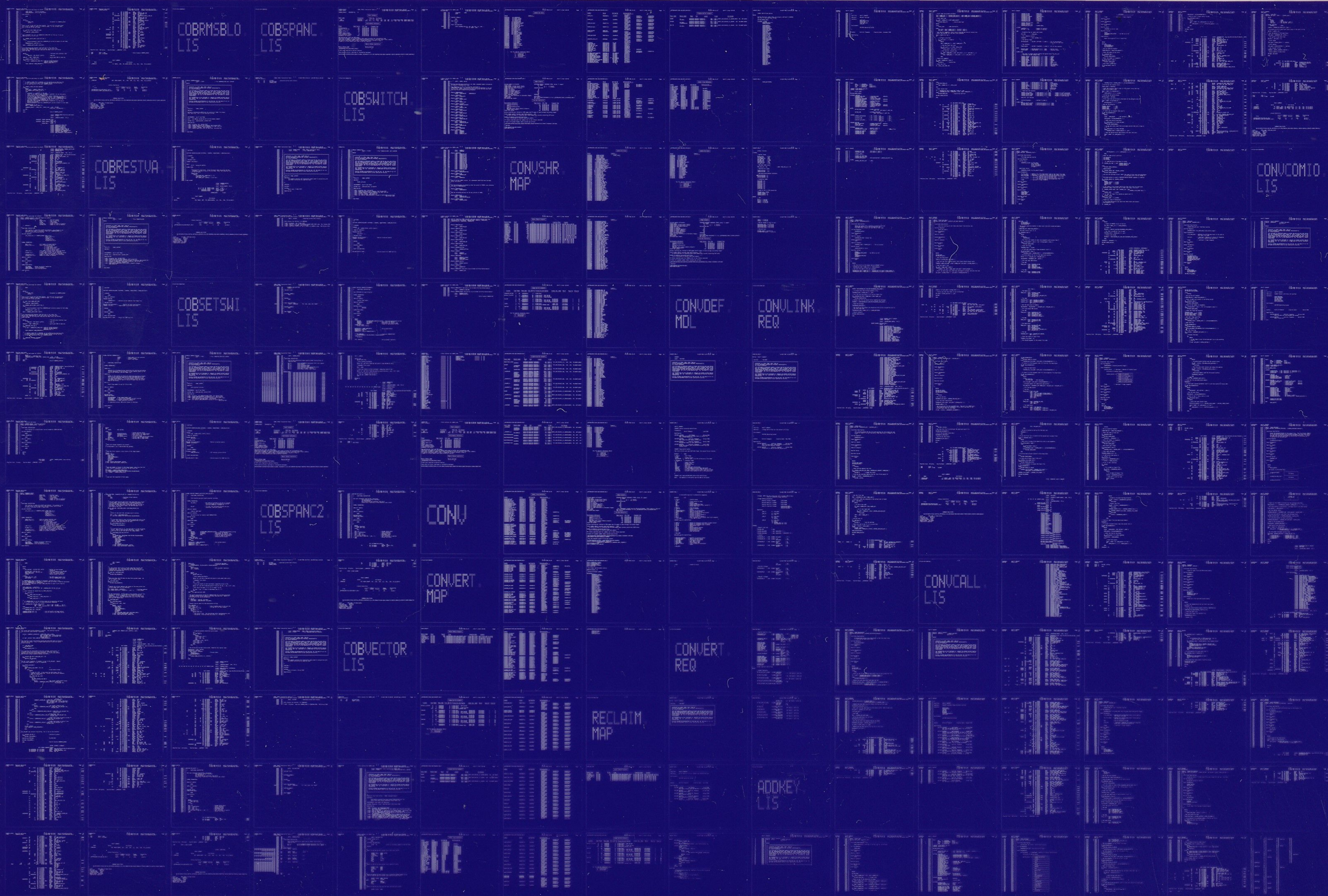
COMMAND QUALIFIERS

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

; Size: 548 code + 12 data bytes
; Run Time: 00:13.1
; Elapsed Time: 00:35.3
; Lines/CPU Min: 3211
; Lexemes/CPU-Min: 10816
; Memory Used: 97 pages
; Compilation Complete

0064 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY