


```
1 0001 0 MODULE FOR$$FMT_INTRP (XTITLE'Fortran Format Statement Interpreter'  
2 0002 0 IDENT = '2-037' . File: FORFMTINT.B32 Edit: SBL2037  
3 0003 0 ) =  
4 0004 1 BEGIN  
5 0005 1  
6 0006 1 *****  
7 0007 1 *  
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *  
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *  
10 0010 1 * ALL RIGHTS RESERVED. *  
11 0011 1 *  
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *  
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *  
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *  
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *  
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *  
17 0017 1 * TRANSFERRED. *  
18 0018 1 *  
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *  
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *  
21 0021 1 * CORPORATION. *  
22 0022 1 *  
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *  
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *  
25 0025 1 *  
26 0026 1 *  
27 0027 1 *****  
28 0028 1  
29 0029 1 ++  
30 0030 1 FACILITY: FORTRAN  
31 0031 1  
32 0032 1 ABSTRACT:  
33 0033 1  
34 0034 1 This module interprets FORTRAN format statements  
35 0035 1 which have been pre-compiled into an encoded form by either the  
36 0036 1 FORTRAN compiler or the run-time format compiler,  
37 0037 1 FOR$$FMT_COMPIL. It is independent of READ and WRITE semantics  
38 0038 1 and is executed at both the READ Formatted and WRITE Formatted  
39 0039 1 User Data Formatters (UDF) level of abstraction.  
40 0040 1  
41 0041 1 AUTHOR: Peter Yuo, CREATION DATE: 17-Feb-77  
42 0042 1  
43 0043 1 MODIFIED BY:  
44 0044 1 Peter Yuo, 25-Feb-77, Version 1  
45 0045 1 01 - Original  
46 0046 1  
47 0047 1 Richard Grove, 19-Aug-77, Version 2  
48 0048 1 [Previous edit history removed. SBL 23-Aug-1982]  
49 0049 1 2-032 - Add defaults for 0 and Z format width when value is not 1, 2, 4, 8 or  
50 0050 1 16 bytes. SBL 29-Dec-1980  
51 0051 1 2-033- Improved fix for 2-032, courtesy of Joel CLinkenbeard. SBL 8-Jan-1981  
52 0052 1 2-034 - Convert FOR$$FMT_INTRP1 to JSB linkage for better performance.  
53 0053 1 JAW 29-Jul-1981  
54 0054 1 2-035 - Miscellaneous performance enhancements: JAW 29-Jul-1981  
55 0055 1 Check for certain specific one-byte format codes at the outset  
56 0056 1 and special-case them.  
57 0057 1 For all format codes, if optional second byte is not present,
```

```
: 58      0058  1  | bypass checks for VFEs and for optional forms of RC and W.  
: 59      0059  1  | Break FI_ACT into two tables, each having 1-byte entries,  
: 60      0060  1  | placing the special action in FI_ACT_2 and indicating the  
: 61      0061  1  | need for special action with the low-order bit of FI_ACT.  
: 62      0062  1  | Select a special action only if this bit is set.  
: 63      0063  1  | For codes DF through DD, check for element size of 4 first.  
: 64      0064  1  | Narrow the scope of ACT, FMT_REPR and P, which are not needed  
: 65      0065  1  | in the outermost block, to conserve registers.  
: 66      0066  1  | Replace CASE on V_RC_TYPE with IF ... THEN to avoid an EXTZV.  
: 67      0067  1  | 2-036 - Correct range check of P value from VFEs. SBL 23-Aug-1982  
: 68      0068  1  | 2-037 - Allow zero-value VFEs for W, D and E fields only. Use prologue  
: 69      0069  1  | file. SBL 26-Apr-1983  
: 70      0070  1  | -
```



```

114 0177 1 GLOBAL ROUTINE FOR$$FMT_INTRP0 ! Format interpreter initialization
115 0178 1 : JSB_FMT0 NOVALUE =
116 0179 1
117 0180 1 !++
118 0181 1
119 0182 1 FUNCTIONAL DESCRIPTION:
120 0183 1
121 0184 1     Initializes the format interpreter
122 0185 1
123 0186 1 IMPLICIT INPUTS:
124 0187 1
125 0188 1     CCB             Contains adr. of current LUB/ISB/RAB.
126 0189 1
127 0190 1 IMPLICIT OUTPUTS:
128 0191 1
129 0192 1     CCB [ISB$W_FMT_REP] Set repeat count to 0 to indicate no repeat
130 0193 1     for this statement yet.
131 0194 1     CCB [ISB$B_FMT_P]   Set P scale factor to 0 for this statement
132 0195 1     CCB [ISB$A_FMT_PTR] Initializes format pointer to
133 0196 1     beginning
134 0197 1     CCB [ISB$W_FMT_REVER] Offset of current format reversion
135 0198 1     point
136 0199 1     CCB [ISB$B_FMT_DEP]  Depth of repeat group pushdown stack
137 0200 1
138 0201 1 SIDE EFFECTS:
139 0202 1
140 0203 1     NONE
141 0204 1
142 0205 1 !--
143 0206 1
144 0207 2 BEGIN
145 0208 2
146 0209 2     EXTERNAL REGISTER
147 0210 2     CCB : REF $FOR$CCB_DECL;
148 0211 2
149 0212 2 !+
150 0213 2 ! Set repeat count to 0 to indicate no repeat for this statement.
151 0214 2 !-
152 0215 2
153 0216 2 CCB [ISB$W_FMT_REP] = 0;
154 0217 2
155 0218 2 !+
156 0219 2 ! Set P scale factor to 0 for this statement (no scaling).
157 0220 2 !-
158 0221 2
159 0222 2 CCB [ISB$B_FMT_P] = 0;
160 0223 2
161 0224 2 !+
162 0225 2 ! Set format flags to zero for this statement.
163 0226 2 !-
164 0227 2
165 0228 2 CCB [ISB$W_FMT_FLAGS] = 0;
166 0229 2
167 0230 2 !+
168 0231 2 ! Set BN flag if LUB$V_NULLBLNK is set
169 0232 2 !-
170 0233 2

```

```

171 0234 2 CCB [ISB$V_BN] = .CCB [LUB$V_NULLBLNK];
172 0235 2
173 0236 2
174 0237 2 | +
175 0238 2 | Set current format position to beginning of format.
176 0239 2 | -
177 0240 2 CCB [ISB$A_FMT_PTR] = .CCB [ISB$A_FMT_BEG];
178 0241 2
179 0242 2 | +
180 0243 2 | Initialize format reversion point to beginning of format
181 0244 2 | byte array. The reversion point is used when there are
182 0245 2 | more user data elements than data format codes.
183 0246 2 | Since it is a 16-bit offset with respect to ISB$A_FMT_BEG, set to 0.
184 0247 2 | -
185 0248 2
186 0249 2 CCB [ISB$W_FMT_REVER] = 0;
187 0250 2
188 0251 2 | +
189 0252 2 | Initialize format repeat group push down stack depth
190 0253 2 | to empty (-1). 0 = 1 item, 1 = 2 items in stack, etc.
191 0254 2 | -
192 0255 2
193 0256 2 CCB [ISB$B_FMT_DEP] = -1;
194 0257 2
195 0258 2 | +
196 0259 2 | Initialize ISB$B_FMT_CODE to zero, which will tell
197 0260 2 | FOR$$UDF_WF9 not to call FOR$$UDF_WF1 unless there were no
198 0261 2 | items in the I/O list.
199 0262 2 | -
200 0263 2
201 0264 2 CCB [ISB$B_FMT_CODE] = 0;
202 0265 2
203 0266 2 | +
204 0267 2 | All other ISB locations and flags have already been
205 0268 2 | initialized to 0 or a specified value by the I/O statement
206 0269 2 | initialization for this I/O statement.
207 0270 2 | -
208 0271 2
209 0272 2 RETURN;
210 0273 1 END;

```

! End of routine FOR\$\$FMT_INTRPO

```

.TITLE FOR$$FMT_INTRP Fortran Format Statement Interpr
eter
.IDENT \2-037\
.EXTRM FOR$$SIGNAL_STO
.EXTRM FOR$$SIGNAL
.PSECT _FOR$CODE,NOWRT, SHR, PIC,2

```

```

8D AB B4 0000 FOR$$FMT_INTRPO::
88 AB 94 0003 CLRW -115(CCB) : 0216
93 AB B4 0006 CLRW -120(CCB) : 0222
06 EF 0009 CLRW -109(CCB) : 0228
50 F0 000F EXTZV #6, #1, -1(CCB), R0 : 0234
INSV R0, #0, #1, -109(CCB) :

```

```

93 50 FF AB 01 00
AB 01

```



```

212 0274 1 GLOBAL ROUTINE FOR$$FMT_INTRP1      | Format interpret until data code
213 0275 1 |      (EL_SIZE,                          | Size in addressable units of data elements
214 0276 1 |                                          | This argument is passed in
215 0277 1 |                                          | external register EL_SIZE.
216 0278 1 |      DT_SEEN)                          | 1 if data transmitter seen
217 0279 1 |                                          | This argument is passed in
218 0280 1 |                                          | external register DT_SEEN.
219 0281 1 |      : JSB_FMT1 NOVALUE =              | Value is format code
220 0282 1 |                                          | Returned in external register
221 0283 1 |                                          | FMT_CODE.
222 0284 1 |
223 0285 1 |      ++
224 0286 1 |
225 0287 1 |      FUNCTIONAL DESCRIPTION:
226 0288 1 |
227 0289 1 |      FOR$$FMT_INTRP1 interprets FORTRAN format statements
228 0290 1 |      which have been precompiled into an encoded form by either
229 0291 1 |      the FORTRAN compiler or the run-time format compiler,
230 0292 1 |      FOR$FMT_COMPIL. Only FOR$$FMT_INTRP1 understands the structure
231 0293 1 |      and encoding of compiled format statements. Furthermore,
232 0294 1 |      it is independent of READ and WRITE semantics.
233 0295 1 |      Each call to FOR$$FMT_INTRP1 processes as many format
234 0296 1 |      codes as possible until it encounters one which
235 0297 1 |      needs to access user program data, needs to access the
236 0298 1 |      data buffer, or depend on whether read or write. This
237 0299 1 |      block is independent of whether a READ or WRITE is
238 0300 1 |      being performed. It is invoked in both the formatted
239 0301 1 |      READ and WRITE user data formatter routines (UDF).
240 0302 1 |
241 0303 1 |      Note: being compatible with -11 OTS, there is a
242 0304 1 |      difference between nf and n(f) in that VFE's are evaluated only
243 0305 1 |      once for the former and each time for the latter. The former
244 0306 1 |      is termed, repeating a format code while the latter is
245 0307 1 |      termed a repeat group.
246 0308 1 |
247 0309 1 |      IMPLICIT INPUTS:
248 0310 1 |
249 0311 1 |      EL_SIZE                            Size in addressable units of user
250 0312 1 |                                          data element. Used to set default
251 0313 1 |                                          widths (W) for default format.
252 0314 1 |                                          A value of 0 indicates that this is
253 0315 1 |                                          the end of the I/O list call and there is
254 0316 1 |                                          no user I/O list element to be transmitted.
255 0317 1 |
256 0318 1 |      DT_SEEN                            1 if a data transmitter has been seen by
257 0319 1 |                                          the current call to the UDF level, 0
258 0320 1 |                                          otherwise. If it is set, we don't evaluate
259 0321 1 |                                          format items for data transmitters.
260 0322 1 |
261 0323 1 |      OTS$$A_CUR_LUB                     Adr. of current logical unit block
262 0324 1 |                                          (LUB) Used to setup base (ISB) to
263 0325 1 |                                          I/O statement block.
264 0326 1 |
265 0327 1 |      The following locations are set only by previous calls to
266 0328 1 |      FOR$$FMT_INTRP1, i.e., are effectively OWN:
267 0329 1 |
268 0330 1 |      CCB [ISB$A_FMT_PTR]               Adr. of next byte in compiled

```

```
269 0331 1 format byte array. A value of 0
270 0332 1 indicates that this is the end of the
271 0333 1 I/O list call and there is no user
272 0334 1 I/O list element to be transmitted.
273 0335 1 CCB [ISBSV_USER_ELEM] 0 until a user element format
274 0336 1 code seen. Infinite loop preventer
275 0337 1 CCB [ISBSW_FMT_REP] Current format code repeat count (n)
276 0338 1 or 0 if not repeating a single
277 0339 1 format code. Note: the repeat
278 0340 1 count for a repeat group is kept
279 0341 1 in the top of the format stack, not here.
280 0342 1
281 0343 1 CCB [ISBSA_FMT_BEG] Adr. of beginning of format statement
282 0344 1 CCB [ISBSB_FMT_DEP] Depth of repeat group format pushdown stack.
283 0345 1 CCB [ISBSW_FMT_STKP] Stack of offsets to beginning of repeat groups
284 0346 1 CCB [ISBSW_FMT_STKR] Stack of group repeat counts
285 0347 1 CCB [ISBSW_FMT_REVER] Offset of current format reversion
286 0348 1 point to revert to when end of format
287 0349 1 statement is encountered with more data
288 0350 1 elements to be transmitted.
289 0351 1 CCB [ISBSV_USER_ELEM] Flag: 1 if seen a user data element format code,
290 0352 1 0 if not. Used to check for infinite format loop
291 0353 1 in which no user data element format codes are present
292 0354 1
293 0355 1 IMPLICIT OUTPUTS:
294 0356 1
295 0357 1 The following are outputs only to a successive call to
296 0358 1 FOR$$FMT_INTRP(0,1), i.e., are effectively OWN.
297 0359 1
298 0360 1 CCB [ISBSV_USER_ELEM] 0 if no user data element format
299 0361 1 code seen this repeat group, 1
300 0362 1 if one or more
301 0363 1 CCB [ISBSW_FMT_REP] Current format code repeat count (n)
302 0364 1 or 0 if not repeating a single
303 0365 1 format code. Note: the repeat
304 0366 1 count for a repeat group is kept
305 0367 1 in the top of the format stack, not here.
306 0368 1 CCB [ISBSB_FMT_DEP] Depth of repeat group format pushdown stack.
307 0369 1 CCB [ISBSW_FMT_STKP] Stack of offsets to beginning of repeat groups
308 0370 1 CCB [ISBSW_FMT_STKR] Stack of group repeat counts
309 0371 1 CCB [ISBSW_FMT_REVER] Offset of current format reversion
310 0372 1 point to revert to when end of format
311 0373 1 statement is encountered with more data
312 0374 1 elements to be transmitted.
313 0375 1
314 0376 1 The following are output to available to the caller (read
315 0377 1 or write user data formatter):
316 0378 1
317 0379 1 CCB [ISBSA_FMT_PTR] Adr. of next byte to be read from
318 0380 1 the compiled format statement byte array
319 0381 1 are pushed as a pair.
320 0382 1 CCB [ISBSB_FMT_P] Signed scale factor (P)
321 0383 1 CCB [ISBSW_FMT_W] Unsigned width of field (W)
322 0384 1 CCB [ISBSB_FMT_D] Unsigned number of digits in fraction (D)
323 0385 1 CCB [ISBSB_FMT_E] Unsigned number of characters
324 0386 1 in exponent (E).
325 0387 1 CCB [ISBSV_USER_ELEM] Flag: 1 if seen a user data element format code.
```



```

336 0397 1 !
337 0398 2 BEGIN
338 0399 2
339 0400 2 EXTERNAL REGISTER
340 0401 2 CCB : REF $FOR$CCB_DECL, ! Pointer to Common Control Block
341 0402 2 EL_SIZE, ! Element size (1st argument)
342 0403 2 DT_SEEN, ! Data transmitter seen (2nd argument)
343 0404 2 FMT_CODE : BLOCK [1, LONG]; ! Format code (return value)
344 0405 2
345 0406 2 BUILTIN
346 0407 2 TESTBITSC;
347 0408 2
348 0409 2 MACRO ! Field definitions for action table
349 0410 2 FI_STOP = 0,6,1,0 %; ! Stop if DT_SEEN
350 0411 2 FI_GETW = 0,5,1,0 %; ! Get w value for format
351 0412 2 FI_GETD = 0,4,1,0 %; ! Get d value for format
352 0413 2 FI_GETE = 0,3,1,0 %; ! Get e value for format
353 0414 2 FI_USER = 0,2,1,0 %; ! Format code involves user data element
354 0415 2 FI_EXIT = 0,1,1,0 %; ! Exit from format interpreter loop
355 0416 2 FI_ACTION = 0,0,1,0 %; ! Code-specific action required;
356 0417 2 ! see FI_ACT_2 for action
357 0418 2
358 0419 2 ! MAINTENANCE NOTE: An optimization
359 0420 2 ! below assumes knowledge of certain
360 0421 2 ! bit settings in FI_ACT. See comments.
361 0422 2
362 0423 2 MACRO
363 0424 2 FI_ALL = ! Enumerate all format codes
364 0425 2 W D E U E S
365 0426 2 S E X T
366 0427 2 E R T P
367 0428 2
368 0429 2 FI_PACK(0,0,0,0,1,1,0), ! ER = 0, 00 ! Format syntax error
369 0430 2 FI_PACK(0,0,0,0,0,0,2), ! LP = 1, 01 ! ( - Format reversin point
370 0431 2 FI_PACK(0,0,0,0,0,0,3), ! NLP = 2, 02 ! n( - Left paren of repeat group
371 0432 2 FI_PACK(0,0,0,0,0,0,4), ! RP = 3, 03 ! ) - Right paren of repeat group
372 0433 2 FI_PACK(0,0,0,0,1,1,5), ! EOF = 4, 04 ! ) - End of format
373 0434 2 FI_PACK(0,0,0,0,1,0,1), ! SLS = 5, 05 ! / - Record separator
374 0435 2 FI_PACK(0,0,0,0,1,0,1), ! DLR = 6, 06 ! $ - Dollar sign: terminal I/O
375 0436 2 FI_PACK(0,0,0,0,1,1,1), ! CLN = 7, 07 ! : - Colon: terminate if end of list
376 0437 2 0, ! UNUSED 8
377 0438 2 FI_PACK(0,0,0,0,0,0,7), ! S = 9, 09 ! S - Make + optional
378 0439 2 FI_PACK(0,0,0,0,0,0,8), ! SP = 10, 0A ! SP - Force optional +
379 0440 2 FI_PACK(0,0,0,0,0,0,7), ! SS = 11, 0B ! SS - Leave out optional +
380 0441 2 FI_PACK(1,0,0,0,0,0,6), ! P = 12, 0C ! sP - signed scale factor
381 0442 2 FI_PACK(1,0,0,0,0,0,11), ! T = 13, 0D ! Tn - Tab Set
382 0443 2 FI_PACK(1,0,0,0,1,0,1), ! X = 14, 0E ! old nX
383 0444 2 FI_PACK(1,0,0,0,1,0,1), ! H = 15, 0F ! nHcccc - Hollerith
384 0445 2 FI_PACK(0,0,0,0,0,0,9), ! BN = 16, 10 ! BN = Blanks are nulls
385 0446 2 FI_PACK(0,0,0,0,0,0,10), ! BZ = 17, 11 ! BZ = Blanks are zeroes
386 0447 2 FI_PACK(1,0,0,0,0,0,12), ! TL = 18, 12 ! TLn = Tab left n columns
387 0448 2 FI_PACK(1,0,0,0,0,0,13), ! TR = 19, 13 ! TRn (new nX) = Tab right n columns
388 0449 2 FI_PACK(0,0,0,1,1,1,1), ! Q = 20, 14 ! Q
389 0450 2 FI_PACK(1,0,0,1,1,1,1), ! A = 21, 15 ! nAw - Alpha numeric
390 0451 2 FI_PACK(1,0,0,1,1,1,1), ! L = 22, 16 ! nLw - Logical
391 0452 2 FI_PACK(1,0,0,1,1,1,1), ! O = 23, 17 ! nOw - Octal
392 0453 2 FI_PACK(1,0,0,1,1,1,1), ! I = 24, 18 ! nIw - Integer

```

```

393 M 0454 2 FI_PACK(1,0,0,1,1,1, 1), : Z = 25, : 19 : nZw - Hexadecimal
394 M 0455 2 FI_PACK(1,1,0,1,1,1, 1), : XO = 26, : 1A : Ow.m Octal
395 M 0456 2 FI_PACK(1,1,0,1,1,1, 1), : XI = 27, : 1B : Iw.m Integer
396 M 0457 2 FI_PACK(1,1,0,1,1,1, 1), : XZ = 28, : 1C : Zw.m Hexadecimal
397 M 0458 2 0, : : UNUSED : 29 :
398 M 0459 2 FI_PACK(1,1,0,1,1,1, 1), : F = 30, : 1E : nFw.d - Fixed format
399 M 0460 2 FI_PACK(1,1,0,1,1,1, 1), : E = 31, : 1F : nEw.d - Scientific notation format
400 M 0461 2 FI_PACK(1,1,0,1,1,1, 1), : G = 32, : 20 : nGw.d - General format
401 M 0462 2 FI_PACK(1,1,0,1,1,1, 1), : D = 33, : 21 : nDw.d - Double Precision format
402 M 0463 2 FI_PACK(1,1,1,1,1,1, 1), : XE = 34, : 22 : rEw.dEe
403 M 0464 2 FI_PACK(1,1,1,1,1,1, 1), : XG = 35, : 23 : nGw.dEe
404 M 0465 2 0,0,0,0,0,0, : : UNUSED : 36:40 :
405 M 0466 2 FI_PACK(0,0,0,1,1,1, 1), : DA = 41, : 29 : nA - default A
406 M 0467 2 FI_PACK(0,0,0,1,1,1, 1), : DL = 42, : 2A : nL - default L
407 M 0468 2 FI_PACK(0,0,0,1,1,1, 1), : DO = 43, : 2B : nO - default O
408 M 0469 2 FI_PACK(0,0,0,1,1,1, 1), : DI = 44, : 2C : nI - default I
409 M 0470 2 FI_PACK(0,0,0,1,1,1, 1), : DZ = 45, : 2D : nZ - default Z
410 M 0471 2 0,0,0,0, : : UNUSED : 46:49 :
411 M 0472 2 FI_PACK(0,0,0,1,1,1, 1), : DF = 50, : 32 : nF - default F
412 M 0473 2 FI_PACK(0,0,0,1,1,1, 1), : DE = 51, : 33 : nE - default E
413 M 0474 2 FI_PACK(0,0,0,1,1,1, 1), : DG = 52, : 34 : nG - default G
414 M 0475 2 FI_PACK(0,0,0,1,1,1, 1), : DD = 53, : 35 : nD - default D
415 M 0476 2 X: : End of macro FI_ALL
416 M 0477 2
417 M 0478 2
418 M 0479 2 :+ Define FI_PACK for use in constructing FI_ACT
419 M 0480 2 :-
420 M 0481 2
421 M 0482 2 MACRO : Attributes-packing macro for attributes table
422 M 0483 2 FI_PACK (W, D, E, U, X, S, NDX) =
423 M 0484 2 (S^6 + W^5 + D^4 + E^3 + U^2 + X^1 +
424 M 0485 2 %IF %IDENTICAL (NDX, 1) %THEN 0 %ELSE 1 %FI) %;
425 M 0486 2
426 M 0487 2 BIND
427 M 0488 2 FI_ACT = ! First action table
428 M 0489 2 UPLIT BYTE ( FI_ALL ) : VECTOR [54, BYTE];
429 M 0490 2
430 M 0491 2 :+
431 M 0492 2 :- Redefine FI_PACK for use in constructing FI_ACT_2
432 M 0493 2
433 M 0494 2
434 M 0495 2 UNDECLARE %QUOTE FI_PACK;
435 M 0496 2
436 M 0497 2 MACRO
437 M 0498 2 FI_PACK (W, D, E, U, X, S, NDX) =
438 M 0499 2 NDX %;
439 M 0500 2
440 M 0501 2 BIND
441 M 0502 2 FI_ACT_2 = ! Second action table
442 M 0503 2 UPLIT BYTE ( FI_ALL ) : VECTOR [54, BYTE];
443 M 0504 2
444 M 0505 2 !<BLF/PAGE>

```

```

: 446 0506 2
: 447 0507 2
: 448 0508 2
: 449 0509 2
: 450 0510 2
: 451 0511 2
: 452 0512 2
: 453 0513 2
: 454 0514 2
: 455 0515 2
: 456 0516 2
: 457 0517 2
: 458 0518 2
: 459 0519 2
: 460 0520 3
: 461 0521 3
: 462 0522 3
: 463 0523 3
: 464 0524 3
: 465 0525 3
: 466 0526 3
: 467 0527 3
: 468 0528 4
: 469 0529 4
: 470 0530 4
: 471 0531 3
: 472 0532 3
: 473 0533 3
: 474 0534 2
: 475 0535 2
: 476 0536 2
: 477 0537 2
: 478 0538 2
: 479 0539 2
: 480 0540 2
: 481 0541 2
: 482 0542 2
: 483 0543 2
: 484 0544 2
: 485 0545 3
: 486 0546 3
: 487 0547 3
: 488 0548 3
: 489 0549 3
: 490 0550 3
: 491 0551 3
: 492 0552 3
: 493 0553 3
: 494 0554 4
: 495 0555 4
: 496 0556 4
: 497 0557 4
: 498 0558 4
: 499 0559 4
: 500 0560 4
: 501 0561 4
: 502 0562 4

```

```

+
(NXT!TM+1)
Assume that a format code is being repeated.- nf not n(f).
(as distinguished from a repeat group which is n(...))
Decrement format repeat count (ISB$W_FMT_REP). Test
if still more to repeat - if yes, skip usual format code
dispatching by skipping loop altogether, redo defaults if
default format codes and RETURN
-
IF .CCB [ISB$W_FMT_REP] GTR 1
THEN
BEGIN
LOCAL
ACT : BLOCK [1, LONG]; ! Action table entry for format code

FMT_CODE = .CCB [ISB$B_FMT_CODE];
ACT = .FI_ACT [.FMT_CODE];
IF .DT_SEEN AND .ACT [FI_STOP]
THEN
BEGIN
FMT_CODE = 0;
RETURN;
END;
CCB [ISB$W_FMT_REP] = .CCB [ISB$W_FMT_REP] - 1;
END
ELSE
+
(FINTRP)
Not in format code repeat - start format interpret loop
Loop until encounter a format code which needs to access
data (ER or explicit or default Q, A, L, O, I, Z, F, E, G, or D),
needs to access the data buffer (X, H, Q), or
depends on whether read or write (, /, $, :, T).
-
BEGIN
REGISTER
P : BLOCK [1, LONG]; ! Pointer to format byte stream
ACT : BLOCK [1, LONG]; ! Action table entry for format code
P = .CCB [ISB$A_FMT_PTR];
DO
BEGIN
+
Pickup next format code byte from compiled format:
If optional representation byte
is present (V_FMT_REPRE=1), mask out flag bit
in format code and copy next byte to BITVECTOR
to indicate larger (less frequent) sizes of the
code representation or Variable Field Expressions (VFE).

```

```

: 503      0563  4      !-
: 504      0564  4
: 505      0565  4      FMT_CODE = CH$RCHAR (.P);
: 506      0566  4      FMT_CODE [V_FMT_REPRE] = 0;          ! Clear bit for search
: 507      0567  4      ACT = .FI_ACT [FMT_CODE];
: 508      0568  4
: 509      0569  4
: 510      0570  4      +
: 511      0571  4      | If DT_SEEN is set and this format code needs a data transmitter
: 512      0572  4      | then return a format code of
: 513      0573  4      | zero to signal the fact. This will be differentiated from
: 514      0574  4      | an error by the UDF level by checking DT_SEEN.
: 515      0575  4      |
: 516      0576  4      |
: 517      0577  4      |
: 518      0578  5      IF .DT_SEEN AND .ACT [FI_STOP]
: 519      0579  5      THEN
: 520      0580  5          BEGIN
: 521      0581  5          CCB [ISB$A_FMT_PTR] = .P;
: 522      0582  4          FMT_CODE = 0;
: 523      0583  4          RETURN;
: 524      0584  4          END;
: 525      0585  4      FMT_CODE = CH$RCHAR_A (P);          ! Re-read and increment pointer
: 526      0586  4
: 527      0587  4      +
: 528      0588  4      | Optimization:
: 529      0589  4      |
: 530      0590  4      | Check for certain easily-handled (and frequent) cases:
: 531      0591  4      | 1. A/L/O/I/Z (codes 21-25) with no RC and byte-length W;
: 532      0592  4      | 2. O/I/Z/F/E/D/G (codes 26-28 and 30-33) with no RC and
: 533      0593  4      |   byte-length W, D;
: 534      0594  4      | 3. E/G (codes 34-35) with no RC and byte-length W, D, E;
: 535      0595  4      | If found, handle directly and bypass the tests for VFE's,
: 536      0596  4      | word-length RC and W, and special action. Note that
: 537      0597  4      | anything with V_FMT_REPRE set falls under OUTRANGE.
: 538      0598  4      |
: 539      0599  4      | This optimization assumes knowledge of flag bit settings
: 540      0600  4      | in FI_ACT, and must be reconsidered if FI_ACT changes.
: 541      0601  4      |
: 542      0602  5      |
: 543      0603  5      |
: 544      0604  5      |
: 545      0605  6      |
: 546      0606  6      |
: 547      0607  6      |
: 548      0608  6      |
: 549      0609  6      |
: 550      0610  5      |
: 551      0611  5      |
: 552      0612  6      |
: 553      0613  6      |
: 554      0614  6      |
: 555      0615  6      |
: 556      0616  6      |
: 557      0617  6      |
: 558      0618  6      |
: 559      0619  5      |

```

```

560      0620  5      [XE TO XG] :
561      0621  6      BEGIN
562      0622  6      CCB [ISBSW_FMT_W] = RBYTE_A (P);
563      0623  6      CCB [ISBSB_FMT_D] = RBYTE_A (P);
564      0624  6      CCB [ISBSB_FMT_E] = RBYTE_A (P);
565      0625  6      CCB [ISBSW_FMT_REP] = 1;
566      0626  6      CCB [ISBSV_USER_ELEM] = 1;
567      0627  6      1 ! Indicate special case found
568      0628  5      END;
569      0629  5      [29, OVRANGE] :
570      0630  5      0; ! Indicate special case not found
571      0631  5      TES)
572      0632  4      THEN
573      0633  5      BEGIN
574      0634  5      +
575      0635  5      | Get RC, W, D and E in the traditional, fully general
576      0636  5      | way, including check for VFE's and alternate forms of
577      0637  5      | W and RC.
578      0638  5      |
579      0639  5      | Optimization:
580      0640  5      |
581      0641  5      | If optional second byte is not present, bypass check
582      0642  5      | for VFE's and alternate forms of W and RC.
583      0643  5      |
584      0644  5      | -
585      0645  5      IF NOT TESTBITSC (FMT_CODE [V_FMT_REPRE])
586      0646  5      THEN
587      0647  5      BEGIN ! Begin short form
588      0648  6      CCB [ISBSW_FMT_REP] = 1;
589      0649  6      IF .ACT [FI_GETW]
590      0650  6      THEN
591      0651  6      BEGIN
592      0652  7      CCB [ISBSW_FMT_W] = RBYTE_A (P);
593      0653  7      IF .ACT [FI_GETD]
594      0654  7      THEN
595      0655  7      BEGIN
596      0656  8      CCB [ISBSB_FMT_D] = RBYTE_A (P);
597      0657  8      IF .ACT [FI_GETE]
598      0658  8      THEN
599      0659  8      CCB [ISBSB_FMT_E] = RBYTE_A (P);
600      0660  8      END;
601      0661  7      END;
602      0662  6      END ! End short form
603      0663  6      ELSE
604      0664  5      BEGIN ! Begin long form
605      0665  6      LOCAL
606      0666  6      FMT_REPR : BLOCK [1, LONG];
607      0667  6      FMT_REPR = RBYTE_A (P);
608      0668  6      +
609      0669  6      | Get repeat count (RC) from format and save in ISBSW_FMT_REP.
610      0670  6      | If repeat count is a VFE (FMT_REPR[V_RC_VFE]=1), get VFE and
611      0671  6      | check for out of range (1:32767).
612      0672  6      | If explicitly represented, get byte or word value.
613      0673  6
614      0674  6
615      0675  6
616      0676  6

```



```

: 617      0677 6      ! Else set repeat count to 1. Possible for left paren
618      0678 6      ! of a repeat group (NLP) or A, L, O, Z, I, F, E, G, D
619      0679 6      ! or default A, L, O, Z, I, F, E, G, D.
620      0680 6      !
621      0681 6      !
622      0682 7      CCB [ISBSW_FMT_REP] = (IF .FMT_REPR [V_RC_VFE]
623      0683 7      THEN
624      0684 8      BEGIN          ! Process RC VFE
625      0685 8      LOCAL
626      0686 8      T;
627      0687 8      T = CALL_VFE (P);
628      0688 8      IF .T GEQU 32768 OR .T EQL 0
629      0689 8      THEN
630      0690 8      BEGIN
631      0691 8      FOR$$SIGNAL (FOR$K VFEVALERR);
632      0692 8      1          ! Force repeat count to 1 on error
633      0693 9      END
634      0694 9      ELSE
635      0695 9      .T          ! Use user supplied value
636      0696 9      END          ! End of RC VFE processing
637      0697 8      ELSE
638      0698 8      ELSE
639      0699 8      ! The following assumes that RC is either 00
640      0700 8      ! (absent), 01 (byte) or 10 (word), and that
641      0701 7      ! it cannot be 11.
642      0702 7      IF .FMT_REPR [V_RC_TYPE_BYTE]
643      0703 7      THEN
644      0704 7      RBYTE_A (P)          ! RC is a byte
645      0705 7      ELSE
646      0706 7      IF .FMT_REPR [V_RC_TYPE_WORD]
647      0707 7      THEN
648      0708 7      RWORD_A (P)          ! RC is a word
649      0709 8      ELSE
650      0710 7      1);          ! RC is absent
651      0711 7      P, T, X, H, A, L, O, I, Z, F, E, G, D:
652      0712 7      Get field width (w) from format and
653      0713 8      set ISBSW_FMT_W. If width field is a
654      0714 7      VFE (V_W_VFE=T), get VFE value and check range;
655      0715 6      if P scale -128 to 127, else (field width w) 0 to 32767.
656      0716 6      If width of field is a byte (V_W_WORD=0), get byte
657      0717 6      else get word. ISBSW_FMT_W is set as a
658      0718 6      zero extended word.
659      0719 6      !
660      0720 6      !
661      0721 6      !
662      0722 6      !
663      0723 6      !
664      0724 6      !
665      0725 6      !
666      0726 6      !
667      0727 6      !
668      0728 6      IF .ACT [FI_GETW]
669      0729 6      THEN
670      0730 7      BEGIN
671      0731 8      CCB [ISBSW_FMT_W] = (IF .FMT_REPR [V_W_VFE] THEN
672      0732 9      BEGIN
673      0733 9

```

```

: 674      0734  9
: 675      0735  9
: 676      0736  9
: 677      0737  9
: 678      0738  9
: 679      0739  9
: 680      0740  9
: 681      0741 10
: 682      0742 10
: 683      0743 10
: 684      0744 10
: 685      0745 11
: 686      0746 11
: 687      0747 11
: 688      0748 11
: 689      0749 11
: 690      0750 10
: 691      0751  9
: 692      0752 10
: 693      0753 10
: 694      0754 10
: 695      0755 10
: 696      0756 11
: 697      0757 11
: 698      0758 11
: 699      0759 11
: 700      0760 11
: 701      0761  9
: 702      0762  9
: 703      0763  9
: 704      0764  9
: 705      0765  7
: 706      0766  7
: 707      0767  7
: 708      0768  7
: 709      0769  7
: 710      0770  7
: 711      0771  7
: 712      0772  7
: 713      0773  7
: 714      0774  7
: 715      0775  7
: 716      0776  7
: 717      0777  7
: 718      0778  8
: 719      0779  9
: 720      0780 10
: 721      0781 10
: 722      0782 10
: 723      0783 10
: 724      0784 10
: 725      0785 10
: 726      0786 10
: 727      0787 10
: 728      0788 10
: 729      0789 11
: 730      0790 11

LOCAL
  T;

  T = CALL_VFE (P);

  IF .FMT_CODE EQL _P
  THEN
    BEGIN
      ! P scale
      IF .T<0,8,1> NEQ .T      ! P between -128 and 127?
      THEN
        BEGIN
          FOR$$SIGNAL (FOR$K_VFEVALERR);
          T = 0
          ! Force P scale to 0
        END
      END
    ELSE
      BEGIN
        ! Else w width of field
        IF .T GEQU 32768
        THEN
          BEGIN
            FOR$$SIGNAL (FOR$K_VFEVALERR);
            T = 1
          END
        END;

        .T
        END
        ! return VFE value
      ELSE IF .FMT_REPR [V_W_WORD] THEN RWORD_A (P) ELSE RBYTE_A (P));

      !+
      ! Get decimal part (d) from format and set
      ! ISB$B_FMT_D. If decimal part is a VFE
      ! (V_D_VFE=T) get VFE and check range (0:32767).
      ! Else get byte from format
      !-
      Set default exponent width to 2.

      IF .ACT [FI_GETD]
      THEN
        BEGIN
          CCB [ISB$B_FMT_D] = (IF .FMT_REPR [V_D_VFE] THEN
          ! VFE
          LOCAL
            T;

            T = CALL_VFF (P);

            IF .T GEQU 32768
            THEN
              BEGIN
                FOR$$SIGNAL (FOR$K_VFEVALERR);
```

```

: 731      0791 11
: 732      0792 11
: 733      0793 10
: 734      0794 10
: 735      0795 10
: 736      0796 10
: 737      0797 8
: 738      0798 8
: 739      0799 8
: 740      0800 8
: 741      0801 8
: 742      0802 8
: 743      0803 8
: 744      0804 8
: 745      0805 8
: 746      0806 8
: 747      0807 8
: 748      0808 9
: 749      0809 10
: 750      0810 11
: 751      0811 11
: 752      0812 11
: 753      0813 11
: 754      0814 11
: 755      0815 11
: 756      0816 11
: 757      0817 11
: 758      0818 11
: 759      0819 12
: 760      0820 12
: 761      0821 12
: 762      0822 12
: 763      0823 11
: 764      0824 11
: 765      0825 11
: 766      0826 11
: 767      0827 9
: 768      0828 8
: 769      0829 8
: 770      0830 7
: 771      0831 7
: 772      0832 6
: 773      0833 6
: 774      0834 5
: 775      0835 5
: 776      0836 5
: 777      0837 5
: 778      0838 5
: 779      0839 5
: 780      0840 5
: 781      0841 5
: 782      0842 5
: 783      0843 5
: 784      0844 5
: 785      0845 5
: 786      0846 5
: 787      0847 5

      1
      EMR
      ELSE
      .T

      END
      ELSE RBYTE_A (P));
      CCB [ISBSB_FMT_E] = 2;

      !+
      ! Get exponent width (e) from format and set
      ! ISBSB_FMT_E. If exponent width is a VFE, check
      ! range (0:255). Else get byte from format.
      !-

      IF .ACT [FI_GETE]
      THEN
      BEGIN
      CCB [ISBSB_FMT_E] = (IF .FMT_REPR [V_E_VFE] THEN
      BEGIN
      .T VFE

      LOCAL
      T;

      T = CALL_VFE (P);

      IF .T GEQU 256
      THEN
      BEGIN
      FOR$$SIGNAL (FOR$K_VFEVALERR);
      1
      END
      ELSE
      .T

      END
      ELSE RBYTE_A (P));
      END;

      END;

      END;

      ! End long form

      !+
      ! For all user data element format codes
      ! (explicit and default Q, A, L, O, I, Z, F, E, G, D):
      ! Set user data element format code
      ! seen in this group, because not in an
      ! infinite format loop invoking for a user
      ! data element format code which doesn't exist.
      !-

      IF .ACT [FI_USER] THEN CCB [ISBSV_USER_ELEM] = 1;

      !+

```

```

: 788      0848 5      ! Dispatch on format code and select appropriate actions:
: 789      0849 5      !-
: 790      0850 5
: 791      0851 5      IF .ACT [FI_ACTION]
: 792      0852 5      THEN
: 793      0853 5      CASE .FI_ACT_2 [.FMT_CODE] FROM 0 TO 13 OF
: 794      0854 5      SET
: 795      0855 5
: 796      0856 5      [0] :
: 797      0857 5          !+
: 798      0858 5          ! ER or undefined format code
: 799      0859 5          ! Bad format: Signal_stop SYNTAX ERROR IN FORMAT (FOR$_SYNERRFOR)
: 800      0860 5          !-
: 801      0861 5
: 802      0862 5
: 803      0863 6      BEGIN
: 804      0864 6      FOR$$SIGNAL_STO (FOR$_SYNERRFOR);
: 805      0865 6      FMT_CODE = 0;
: 806      0866 6      RETURN;
: 807      0867 5      END;
: 808      0868 5
: 809      0869 5      [1] :
: 810      0870 5
: 811      0871 5          !+
: 812      0872 5          ! No special actions required.
: 813      0873 5          !-
: 814      0874 5
: 815      0875 5
: 816      0876 5
: 817      0877 5      [2] :
: 818      0878 5
: 819      0879 5          !+
: 820      0880 5          ! LP Format reversion point: left paren of
: 821      0881 5          ! second outer-most pair. Remeber current format
: 822      0882 5          ! offset (ISB$_FMT_REVER) in case more data
: 823      0883 5          ! element in I/O list than data format
: 824      0884 5          ! codes in format. Reset push down stack to
: 825      0885 5          ! empty (-1) since this is start of
: 826      0886 5          ! first group repeat. Clear user data element
: 827      0887 5          ! seen flag (ISB$_USER_ELEM) as a defense
: 828      0888 5          ! against infinite loop with no data
: 829      0889 5          ! transmit format code
: 830      0890 5          ! Note: format text pointer already advanced to next byte
: 831      0891 5          !-
: 832      0892 5
: 833      0893 6      BEGIN
: 834      0894 6      CCB [ISB$_FMT_DEP] = -1;
: 835      0895 6      CCB [ISB$_FMT_REVER] = .P - .CCB [ISB$_FMT_BEG];
: 836      0896 6      CCB [ISB$_USER_ELEM] = 0;
: 837      0897 5      END;
: 838      0898 5          ! End LP
: 839      0899 5
: 840      0900 5      [3] :
: 841      0901 5
: 842      0902 5          !+
: 843      0903 5          ! NLP Left paren of a repeat group: Push repeat
: 844      0904 5          ! count (ISB$_FMT_REP) and current (ISB$_FMT_PTR)
:          ! onto format stacks

```

```

: 845      0905      5      !-
: 846      0906      5
: 847      0907      6
: 848      0908      6      BEGIN
: 849      0909      6      CCB [ISB$B_FMT_DEP] = .CCB [ISB$B_FMT_DEP] + 1;
: 850      0910      6      VECTOR [CCB [ISB$W_FMT_STKR], .CCB [ISB$B_FMT_DEP];, WORD, UNSIGNED] !
: 851      0911      6      = .CCB [ISB$W_FMT_REP];
: 852      0912      6      VECTOR [CCB [ISB$W_FMT_STKP], .CCB [ISB$B_FMT_DEP];, WORD, UNSIGNED] !
: 853      0913      6      = .P - .CCB [ISB$A_FMT_BEG];
: 854      0914      5      CCB [ISB$W_FMT_REP] = T;
: 855      0915      5      END;
: 856      0916      5      ! End NLP
: 857      0917      5      [4] :
: 858      0918      5      !+
: 859      0919      5      RP Right paren of repeat group: Decrement
: 860      0920      5      current group repeat count (on top of
: 861      0921      5      stack) If current group count still greater
: 862      0922      5      than 0, set current format pointer back to
: 863      0923      5      beginning of repeat group. Else pop off
: 864      0924      5      beginning of group pointer and group repeat count
: 865      0925      5      !-
: 866      0926      5
: 867      0927      6      IF (VECTOR [CCB [ISB$W_FMT_STKR], .CCB [ISB$B_FMT_DEP];, WORD, UNSIGNED]
: 868      0928      5      = .VECTOR [CCB [ISB$W_FMT_STKR], .CCB [ISB$B_FMT_DEP];, WORD, UNSIGNED] - 1) GTR !
: 869      0929      5      THEN
: 870      0930      5      ! reset pointer to address of repeat group
: 871      0931      5      !
: 872      0932      5      P = .CCB [ISB$A_FMT_BEG] !
: 873      0933      5      + .VECTOR [CCB [ISB$W_FMT_STKP], .CCB [ISB$B_FMT_DEP];, WORD, UNSIGNED]
: 874      0934      5      ELSE
: 875      0935      5      ! pop off pointer and repeat count
: 876      0936      5      !
: 877      0937      5      CCB [ISB$B_FMT_DEP] = .CCB [ISB$B_FMT_DEP] - 1;
: 878      0938      5
: 879      0939      5      [5] :
: 880      0940      5
: 881      0941      5      !+
: 882      0942      5      EOF End of format:
: 883      0943      5      ! If not end of user I/O list (EL_SIZE=0)
: 884      0944      5      ! and no user data element format code
: 885      0945      5      ! (ISB$V_USER_ELEM=0), then Signal_stop. INFINITE
: 886      0946      5      ! FORMAT_LOOP (FOR$_INFFORLOP).
: 887      0947      5      ! Reset current format pointer to reversion point
: 888      0948      5      ! (ISB$W_FMT_REVER). Initialize format stack depth.
: 889      0949      5      !-
: 890      0950      5
: 891      0951      6      BEGIN
: 892      0952      6      P = .CCB [ISB$A_FMT_BEG] + .CCB [ISB$W_FMT_REVER];
: 893      0953      6      CCB [ISB$B_FMT_DEP] = -1;
: 894      0954      6
: 895      0955      6      IF .EL_SIZE GTRU 0 AND NOT .CCB [ISB$V_USER_ELEM]
: 896      0956      6      THEN
: 897      0957      7      BEGIN
: 898      0958      7      FOR$$SIGNAL STO (FOR$K_INFFORLOO);
: 899      0959      7      FMT CODE = 0;
: 900      0960      7      RETURN;
: 901      0961      6      END;

```

```

... 902 0962 6
... 903 0963 5
... 904 0964 5
... 905 0965 5
... 906 0966 5
... 907 0967 5
... 908 0968 5
... 909 0969 5
... 910 0970 5
... 911 0971 5
... 912 0972 5
... 913 0973 6
... 914 0974 6
... 915 0975 5
... 916 0976 5
... 917 0977 5
... 918 0978 5
... 919 0979 5
... 920 0980 5
... 921 0981 5
... 922 0982 5
... 923 0983 6
... 924 0984 6
... 925 0985 5
... 926 0986 5
... 927 0987 5
... 928 0988 5
... 929 0989 5
... 930 0990 5
... 931 0991 5
... 932 0992 5
... 933 0993 6
... 934 0994 6
... 935 0995 5
... 936 0996 5
... 937 0997 5
... 938 0998 5
... 939 0999 5
... 940 1000 5
... 941 1001 5
... 942 1002 5
... 943 1003 6
... 944 1004 6
... 945 1005 5
... 946 1006 5
... 947 1007 5
... 948 1008 5
... 949 1009 5
... 950 1010 5
... 951 1011 5
... 952 1012 5
... 953 1013 6
... 954 1014 6
... 955 1015 5
... 956 1016 5
... 957 1017 5
... 958 1018 5

      END;

[6] :
      !+
      ! P Scale factor (sP): -128 =< s =< 127
      ! Convert unsigned word width (w) (ISB$W_FMT_W)
      ! to signed byte ('s) and save in ISB$B_FMT_P.
      !-

      BEGIN
      CCB [ISB$B_FMT_P] = .CCB [ISB$W_FMT_W];
      END;

[7] :
      !+
      ! S, SS Restore option of + to processor.
      !-

      BEGIN
      CCB [ISB$V_SP] = 0;
      END;

[8] :
      !+
      ! SP Force optional + to appear
      !-

      BEGIN
      CCB [ISB$V_SP] = 1;
      END;

[9] :
      !+
      ! BN Treat blanks as nulls on numeric input.
      !-

      BEGIN
      CCB [ISB$V_BN] = 1;
      END;

[10] :
      !+
      ! BZ Treat blanks as zeroes on numeric input.
      !-

      BEGIN
      CCB [ISB$V_BN] = 0;
      END;

[11] :

```

```

959      1019      5
960      1020
961      1021      5
962      1022      5
963      1023      5
964      1024      6
965      1025      6
966      1026      5
967      1027      5
968      1028      5
969      1029      5
970      1030      5
971      1031      5
972      1032      5
973      1033      5
974      1034      6
975      1035      6
976      1036      6
977      1037      6
978      1038      6
979      1039      6
980      1040      6
981      1041      5
982      1042      5
983      1043      5
984      1044      5
985      1045      5
986      1046      5
987      1047      5
988      1048      5
989      1049      5
990      1050      5
991      1051      5
992      1052      5
993      1053      6
994      1054      6
995      1055      5
996      1056      5
997      1057      5
998      1058      5
999      1059      5
1000     1060      5
1001     1061      5
1002     1062      5
1003     1063      5
1004     1064      5
1005     1065      5
1006     1066      5
1007     1067      5
1008     1068      5
1009     1069      5
1010     1070      5
1011     1071      5
1012     1072      4
1013     1073      3
1014     1074      3
1015     1075      3

```

```

      !+
      |  In  Move buffer pointer to position n
      | -
      |
      | BEGIN
      | CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_BEG] + (.CCB [ISB$W_FMT_W] - 1);
      | END;
[12] :
      |
      | !+
      | |  TLn  Move buffer pointer left n positions
      | | -
      | |
      | | BEGIN
      | | CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] - .CCB [ISB$W_FMT_W];
      | | IF .CCB [LUB$A_BUF_PTR] LSSA .CCB [LUB$A_BUF_BEG]
      | | THEN
      | |   CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_BEG];
      | | END;
[13] :
      |
      | !+
      | |  TRn  Move buffer pointer right n spaces.
      | |      Note: as of VMS Release 2, the format nX
      | |      is equivalent to TRn. The old nX code
      | |      is no longer generated but is supported
      | |      for compatibility.
      | | -
      | |
      | | BEGIN
      | | CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + .CCB [ISB$W_FMT_W];
      | | END;
      | TES;
      |
      | !+
      | |  End of loop - continue if just format control
      | |  ( (, n(, )) or not dependent on read/write
      | |  and doesn't access data buffer (P)
      | |
      | |  EXITLOOP for format codes which access user data.
      | |  (ER or explicit or default A, L, O, I, Z, F, E, G or D),
      | |  EXITLOOP for format codes which access data
      | |  buffer (X, H, Q) EXITLOOP for format codes
      | |  which depend on whether read or write (end
      | |  of format, /, $, :, T).
      | | -
      | |
      | | END
      | | UNTIL .ACT [FI_EXIT];
      | | !+

```

```

: 1016      1076  3      ! Reset format code and updated format text pointer in ISB.
: 1017      1077  3      !
: 1018      1078  3
: 1019      1079  3      CCB [ISBSB_FMT_CODE] = .FMT_CODE;
: 1020      1080  3      CCB [ISBSA_FMT_PTR] = .P;
: 1021      1081  2
: 1022      1082  2      END;
```



```
1024 1083 2
1025 1084 2
1026 1085 2
1027 1086 2
1028 1087 2
1029 1088 2
1030 1089 2
1031 1090 2
1032 1091 2
1033 1092 2
1034 1093 2
1035 1094 2
1036 1095 2
1037 1096 2
1038 1097 2
1039 1098 2
1040 1099 2
1041 1100 2
1042 1101 2
1043 1102 2
1044 1103 2
1045 1104 2
1046 1105 2
1047 1106 2
1048 1107 2
1049 1108 2
1050 1109 2
1051 1110 2
1052 1111 2
1053 1112 2
1054 1113 2
1055 1114 2
1056 1115 2
1057 1116 2
1058 1117 2
1059 1118 2
1060 1119 2
1061 1120 2
1062 1121 2
1063 1122 2
1064 1123 2
1065 1124 2
1066 1125 2
1067 1126 2
1068 1127 2
1069 1128 2
1070 1129 2
1071 1130 2
1072 1131 2
1073 1132 2
1074 1133 2
1075 1134 2
1076 1135 2
1077 1136 2
1078 1137 2
1079 1138 2
1080 1139 2

+
Default data format codes - set defaults based on size of
each user data element even if inside a format code repeat
since the size could be different for each user data element
-

IF .FMT_CODE GEQU _DA
THEN
  BEGIN
    CASE .FMT_CODE FROM _DA TO _DD OF
      SET
        [_DA] :
          +
          Default A: set w field (ISBSW_FMT_W) from
          size of user data element
          -
          CCB [ISBSW_FMT_W] = .EL_SIZE;
        [_DL] :
          +
          Default L: set w field (ISBSW_FMT_W) to 2
          -
          CCB [ISBSW_FMT_W] = 2;
        [_DI] :
          +
          Default I: Set w field to 7 if element is smaller than
          4 bytes else set it to 12.
          -
          IF .EL_SIZE LSSU 4 THEN CCB [ISBSW_FMT_W] = 7 ELSE CCB [ISBSW_FMT_W] = 12;
        [_DO, _DZ] :
          +
          Default O, Z. Set to the width that would allow O
          format plus a space. \\ Note: For compatibility with
          previous releases, the sizes for 1, 2 and 4 bytes must
          be 7, 7 and 12 respectively. \\
          -
          CCB [ISBSW_FMT_W] = MAX (7, MIN (65535, (((8*.EL_SIZE)+2)/3)+1));
        [_DF, _DE, _DG, _DD] :
          +
          Default F, E, G, D: Set w and e fields as is appropriate
          to the element size. Note that anything that is not
          8 (REAL*8) or 16 (REAL*16) is assumed to be 4 (REAL*4),
          but check for 4 first.
          -
```

```

1081      1140      3
1082      1141      4          BEGIN
1083      1142      4
1084      1143      4          SELECTONE .EL_SIZE OF
1085      1144      4          SET
1086      1145      4
1087      1146      4          [4] :
1088      1147      5          BEGIN
1089      1148      5          CCB [ISBSB_FMT_E] = 2;
1090      1149      5          CCB [ISBSW_FMT_W] = 15;
1091      1150      5          CCB [ISBSB_FMT_D] = 7;
1092      1151      4          END;
1093      1152      4
1094      1153      4          [8] :
1095      1154      5          BEGIN
1096      1155      5          CCB [ISBSB_FMT_E] = 2;
1097      1156      5          CCB [ISBSW_FMT_W] = 25;
1098      1157      5          CCB [ISBSB_FMT_D] = 16;
1099      1158      4          END;
1100      1159      4
1101      1160      4          [16] :
1102      1161      5          BEGIN
1103      1162      5          CCB [ISBSB_FMT_E] = 3;
1104      1163      5          CCB [ISBSW_FMT_W] = 42;
1105      1164      5          CCB [ISBSB_FMT_D] = 33;
1106      1165      4          END;
1107      1166      4
1108      1167      4          [OTHERWISE] :
1109      1168      5          BEGIN
1110      1169      5          CCB [ISBSB_FMT_E] = 2;
1111      1170      5          CCB [ISBSW_FMT_W] = 15;
1112      1171      5          CCB [ISBSB_FMT_D] = 7;
1113      1172      4          END;
1114      1173      4          TES;
1115      1174      4
1116      1175      3          END;
1117      1176      3
1118      1177      3          [INRANGE] :
1119      1178      3          TES;
1120      1179      3
1121      1180      3
1122      1181      3          !+
1123      1182      3          ! Translate default format code to corresponding explicit code.
1124      1183      3          !-
1125      1184      3
1126      1185      3          FMT_CODE = .FMT_CODE - (_DA - _A);
1127      1186      2          END;
1128      1187      2
1129      1188      2
1130      1189      2          !+
1131      1190      2          ! Return to read, write User Data Formatter (UDF). If default
1132      1191      2          ! format code, return corresponding explicit format code
1133      1192      2          ! to UDF. Else return the actual format code
1134      1193      2          !-
1135      1194      2          RETURN;
1136      1195      2
1137      1196      1          END;

```

! End of routine FOR\$\$FMT_INTRP1

FC
SY
FC
SY
PS
--
_F
Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As
Th
16
Th
13
2
Ma
--
_S
13
Th
MA

	89	AB		82	9B	0006B	6\$:	MOVZBW	(P)+, -119(CCB)	:	0606
				16	11	0006F		BRB	9\$:	0607
	89	AB		82	9B	00071	7\$:	MOVZBW	(P)+, -119(CCB)	:	0613
	8B	AB		82	90	00075		MOVB	(P)+, -117(CCB)	:	0614
	8C	AB		02	90	00079		MOVB	#2, -116(CCB)	:	0615
				08	11	0007D		BRB	9\$:	0616
	89	AB		82	9B	0007F	8\$:	MOVZBW	(P)+, -119(CCB)	:	0622
	8B	AB		82	B0	00083		MOVW	(P)+, -117(CCB)	:	0623
	00	BE		01	B0	00087	9\$:	MOVW	#1, @0(SP)	:	0625
	96	AB		08	88	00088		BISB2	#8, -106(CCB)	:	0626
				0236	31	0008F		BRW	50\$:	
28		58		07	E4	00092	10\$:	BBSC	#7, FMT_CODE, 14\$:	0646
	00	BE		01	B0	00096		MOVW	#1, @0(SP)	:	0649
03		53		05	E0	0009A		BBS	#5, ACT, 11\$:	0650
				012A	31	0009E		BRW	30\$:	
	89	AB		82	9B	000A1	11\$:	MOVZBW	(P)+, -119(CCB)	:	0653
03		53		04	E0	000A5		BBS	#4, ACT, 12\$:	0654
				011F	31	000A9		BRW	30\$:	
	8B	AB		82	90	000AC	12\$:	MOVB	(P)+, -117(CCB)	:	0657
03		53		03	E0	000B0		BBS	#3, ACT, 13\$:	0658
				0114	31	000B4		BRW	30\$:	
	8C	AB		82	90	000B7	13\$:	MOVB	(P)+, -116(CCB)	:	0660
				010D	31	000BB		BRW	30\$:	0646
	08	AE		82	9A	000BE	14\$:	MOVZBL	(P)+, FMT_REPR	:	0670
			08	AE	95	000C2		TSTB	FMT_REPR	:	0682
				21	18	000C5		BGEQ	16\$:	
		50		82	D0	000C7		MOVL	(P)+, T	:	0689
	00008000	6240		00	FB	000CA		CALLS	#0, (P)[T]	:	
		8F		50	D1	000CE		CMPL	T, #32768	:	0691
				04	1E	000D5		BGEQU	15\$:	
				50	D5	000D7		TSTL	T	:	
				23	12	000D9		BNEQ	19\$:	
		7E		8F	9A	000DB	15\$:	MOVZBL	#68, -(SP)	:	0694
	00000000G	00		01	FB	000DF		CALLS	#1, FOR\$\$SIGNAL	:	
				13	11	000E6		BRB	18\$:	0693
		05		08	AE	000E8	16\$:	BLBC	FMT_REPR, 17\$:	0707
		50		82	9A	000EC		MOVZBL	(P)+, R0	:	0709
				0D	11	000EF		BRB	19\$:	
05		08	AE	01	E1	000F1	17\$:	BBC	#1, FMT_REPR, 18\$:	0711
		50		82	3C	000F6		MOVZWL	(P)+, R0	:	0713
				03	11	000F9		BRB	19\$:	
		50		01	D0	000FB	18\$:	MOVL	#1, R0	:	0711
		00	BE	50	B0	000FE	19\$:	MOVW	R0, @0(SP)	:	0682
03		53		05	E0	00102		BBS	#5, ACT, 20\$:	0728
				00C2	31	00106		BRW	30\$:	
48		08	AE	06	E1	00109	20\$:	BBC	#6, FMT_REPR, 23\$:	0731
		50		82	D0	0010E		MOVL	(P)+, T	:	0737
		6240		00	FB	00111		CALLS	#0, (P)[T]	:	
		04	AE	50	D0	00115		MOVL	R0, T	:	
		0C		58	D1	00119		CMPL	FMT_CODE, #12	:	0739
				19	12	0011C		BNEQ	21\$:	
04	AE		04	00	EC	0011E		CMPV	#0, #8, T, T	:	0743
				29	13	00125		BEQL	22\$:	
		7E		44	8F	9A	00127	MOVZBL	#68, -(SP)	:	0746
	00000000G	00		01	FB	0012B		CALLS	#1, FOR\$\$SIGNAL	:	
				04	AE	00132		CLRL	T	:	0747
				19	11	00135		BRB	22\$:	0741

	00008000	8F	04	AE	D1	00137	21\$:	CMPL	T, #32768	:	0754
		7E	44	OF	1F	0013F		BLSSU	22\$:	
	00000000G	00		8F	9A	00141		MOVZBL	#68, -(SP)	:	0757
	04	AE		01	FB	00145		CALLS	#1, FOR\$\$SIGNAL	:	
		50	04	01	DO	0014C		MOVL	#1, T	:	0758
				AE	DO	00150	22\$:	MOVL	T, R0	:	0763
05	08	AE		0D	11	00154		BRB	25\$:	
		50		02	E1	00156	23\$:	BBC	#2, FMT_REPR, 24\$:	0765
				82	3C	0015B		MOVZWL	(P)+, R0	:	
				03	11	0015E		BRB	25\$:	
		50		82	9A	00160	24\$:	MOVZBL	(P)+, R0	:	
60	89	AB		50	B0	00163	25\$:	MOVW	R0, -119(CCB)	:	0731
20		53		04	E1	00167		BBC	#4, ACT, 30\$:	0776
	08	AE		05	E1	0016B		BBC	#5, FMT_REPR, 26\$:	0779
		50		82	DO	00170		MOVL	(P)+, T	:	0785
		6240		00	FB	00173		CALLS	#0, (P)[T]	:	
	00008000	8F		50	D1	00177		CMPL	T, #32768	:	0787
				13	1F	0017E		BLSSU	27\$:	
		7E	44	8F	9A	00180		MOVZBL	#68, -(SP)	:	0790
	00000000G	00		01	FB	00184		CALLS	#1, FOR\$\$SIGNAL	:	
		50		01	DO	0018B		MOVL	#1, R0	:	0789
				03	11	0018E		BRB	27\$:	0787
		50		82	9A	00190	26\$:	MOVZBL	(P)+, R0	:	0797
	8B	AB		50	90	00193	27\$:	MOVB	R0, -117(CCB)	:	0779
	8C	AB		02	90	00197		MOVB	#2, -116(CCB)	:	0798
2C		53		03	E1	0019B		BBC	#3, ACT, 30\$:	0806
20	08	AE		04	E1	0019F		BBC	#4, FMT_REPR, 28\$:	0809
		50		82	DO	001A4		MOVL	(P)+, T	:	0815
		6240		00	FB	001A7		CALLS	#0, (P)[T]	:	
	00000100	8F		50	D1	001AB		CMPL	T, #256	:	0817
				13	1F	001B2		BLSSU	29\$:	
		7E	44	8F	9A	001B4		MOVZBL	#68, -(SP)	:	0820
	00000000G	00		01	FB	001B8		CALLS	#1, FOR\$\$SIGNAL	:	
		50		01	DO	001BF		MOVL	#1, R0	:	0819
				03	11	001C2		BRB	29\$:	0817
		50		82	9A	001C4	28\$:	MOVZBL	(P)+, R0	:	0827
	8C	AB		50	90	001C7	29\$:	MOVB	R0, -116(CCB)	:	0809
04		53		02	E1	001CB	30\$:	BBC	#2, ACT, 31\$:	0845
	96	AB		08	88	001CF		BISB2	#8, -106(CCB)	:	
		03		53	EB	001D3	31\$:	BLBS	ACT, 32\$:	0851
				00EF	31	001D6		BRW	50\$:	
	0D	00	FDEC	CF48	8F	001D9	32\$:	CASEB	FI ACT 2[FMT_CODE], #0, #13	:	0853
0032	0020	00E8		001C		001E0	33\$:	.WORD	34\$-33\$,-	:	
00A3	009C	0078		0050		001E8			50\$-33\$,-	:	
00BB	00B5	00AF		00A9		001F0			35\$-33\$,-	:	
		00E0		00CA		001F8			36\$-33\$,-	:	
									37\$-33\$,-	:	
									39\$-33\$,-	:	
									42\$-33\$,-	:	
									43\$-33\$,-	:	
									44\$-33\$,-	:	
									45\$-33\$,-	:	
									46\$-33\$,-	:	
									47\$-33\$,-	:	
									48\$-33\$,-	:	
									49\$-33\$:	
			3E	DD	001FC	34\$:		PUSHL	#62	:	0864

			70	11	001FE		BRB	40\$		
		92	AB	01	8E 00200	35\$:	MNEGB	#1, -110(CCB)		0894
90	AB	52		CB	A3 00204		SUBW3	-132(CCB), P, -112(CCB)		0895
		96	AB	08	8A 0020B		BICB2	#8, -106(CCB)		0896
				00B6	31 0020F		BRW	50\$		0853
				92	AB 96 00212	36\$:	INCB	-110(CCB)		0908
		50		92	AB 9A 00215		MOVZBL	-110(CCB), R0		0909
		FF60	CB40	00	BE B0 00219		MOVW	@0(SP), -160(CCB)[R0]		0910
FF50	CB40	52		FF7C	CB A3 00220		SUBW3	-132(CCB), P, -176(CCB)[R0]		0912
		00	BE	01	B0 00229		MOVW	#1, @0(SP)		0913
				0098	31 0022D		BRW	50\$		0853
		50		92	AB 9A 00230	37\$:	MOVZBL	-110(CCB), R0		0927
		51		FF60	CB40 3C 00234		MOVZWL	-160(CCB)[R0], R1		0928
				51	D7 0023A		DECL	R1		
		FF60	CB40	51	B0 0023C		MOVW	R1, -160(CCB)[R0]		
				51	D5 00242		TSTL	R1		
				0D	15 00244		BLEQ	38\$		
		52		FF50	CB40 3C 00246		MOVZWL	-176(CCB)[R0], P		0933
		52		FF7C	CB C0 0024C		ADDL2	-132(CCB), P		
				75	11 00251		BRB	50\$		0932
				92	AB 97 00253	38\$:	DECB	-110(CCB)		0937
				70	11 00256		BRB	50\$		0927
		52		90	AB 3C 00258	39\$:	MOVZWL	-112(CCB), P		0952
		52		FF7C	CB C0 0025C		ADDL2	-132(CCB), P		
		92	AB	01	8E 00261		MNEGB	#1, -110(CCB)		0953
				5A	D5 00265		TSTL	EL SIZE		0955
				5F	13 00267		BEQL	50\$		
5A		96	AB	03	E0 00269		BBS	#3, -106(CCB), 50\$		
				3C	DD 0026E		PUSHL	#60		0958
		00000000G	00	01	FB 00270	40\$:	CALLS	#1, FOR\$\$SIGNAL_STO		
				58	D4 00277	41\$:	CLRL	FMT_CODE		0959
				00F4	31 00279		BRW	66\$		0957
		88	AB	89	AB 90 0027C	42\$:	MOVW	-119(CCB), -120(CCB)		0974
				45	11 00281		BRB	50\$		0853
		94	AB	01	8A 00283	43\$:	BICB2	#1, -108(CCB)		0984
				3F	11 00287		BRB	50\$		0853
		94	AB	01	88 00289	44\$:	BISB2	#1, -108(CCB)		0994
				39	11 0028D		BRB	50\$		0853
		93	AB	01	88 0028F	45\$:	BISB2	#1, -109(CCB)		1004
				33	11 00293		BRB	50\$		0853
		93	AB	01	8A 00295	46\$:	BICB2	#1, -109(CCB)		1014
				2D	11 00299		BRB	50\$		0853
		50		89	AB 3C 0029B	47\$:	MOVZWL	-119(CCB), R0		1025
		50		BC	AB C0 0029F		ADDL2	-68(CCB), R0		
		B0	AB	FF	A0 9E 002A3		MOVAB	-1(R0), -80(CCB)		
				1E	11 002A8		BRB	50\$		0853
		50		89	AB 3C 002AA	48\$:	MOVZWL	-119(CCB), R0		1035
		B0	AB	50	C2 002AE		SUBL2	R0, -80(CCB)		
		BC	AB	B0	AB D1 002B2		CMPL	-80(CCB), -68(CCB)		1037
				0F	1E 002B7		BGEQU	50\$		
		B0	AB	BC	AB D0 002B9		MOVL	-68(CCB), -80(CCB)		1039
				08	11 002BE		BRB	50\$		0853
		50		89	AB 3C 002C0	49\$:	MOVZWL	-119(CCB), R0		1054
		B0	AB	50	C0 002C4		ADDL2	R0, -80(CCB)		
03		53		01	E0 002C8	50\$:	BBS	#1, ACT, 51\$		1073
				FD5A	31 002CC		BRW	3\$		
		8F	AB	58	90 002CF	51\$:	MOVW	FMT_CODE, -113(CCB)		1079

: 1138 1197 1
: 1139 1198 1 END
: 1140 1199 1
: 1141 1200 0 ELUDOM

! End of module FOR\$\$FMT_INTRP

PSECT SUMMARY

Name Bytes Attributes
_FOR\$CODE 1030 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]STARLET.L32;1	9776	0	0	581	00:01.1
_\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	211	29	52	00:00.6
_\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

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

: Size: 922 code + 108 data bytes
: Run Time: 00:29.5
: Elapsed Time: 01:23.3
: Lines/CPU Min: 2441
: Lexemes/CPU-Min: 23045
: Memory Used: 386 pages
: Compilation Complete

