


```

FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF RRRRRRRR LL
FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF RRRRRRRR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF RRRRRRRR LL
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF RRRRRRRR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 00 00 RR RR UU UU DD DD FF RR RR LL
FF 000000 RR RR UUUUUUUUU DDDDDDDD FF RR RR LLLLLLLLLL .....
FF 000000 RR RR UUUUUUUUU DDDDDDDD FF RR RR LLLLLLLLLL .....

```

```

LL 111111 SSSSSSSS
LL 111111 SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL 111111 SSSSSSSS
LLLLLLLLLL 111111 SSSSSSSS

```

```

1 0001 0 MODULE FOR$$UDF_RL (%TITLE'FORTRAN List-directed input, UDF level'
2 0002 0   -IDENT = '1-025'   . File: FORUDFRL.B32 Edit: SBL1025
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 **
31 0031 1 FACILITY:   FORTRAN support library - not user callable
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1   This module implements FORTRAN read list-directed I/O statement
36 0036 1   at the UDF level of abstraction. This module calls the list-
37 0037 1   directed record routines at the record level to read a record.
38 0038 1
39 0039 1 ENVIRONMENT: User access mode, reentrant AST level or not
40 0040 1
41 0041 1 AUTHOR: Jonathan M. Taylor,   CREATION DATE: 5-SEP-77
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 [Previous edit history deleted. SBL 9-June-1981]
46 0046 1 1-001 - Update version number and copyright notice. JBS 16-NOV-78
47 0047 1 1-002 - Make SKIPBLANKS return a value to keep the BLISS
48 0048 1   compiler happy. JBS 27-NOV-78
49 0049 1 1-003 - Change REQUIRE file names from FOR... to OTS... JBS 06-DEC-78
50 0050 1 1-004 - Change module name to FOR$$UDF_RL to agree with file name. JBS 11-DEC-78
51 0051 1 1-005 - Change ISB$A_BUF_PTR, BUF_END, BUF_BEG, BUF_HIGH to LUB. DGP 08-Jan-79
52 0052 1 1-006 - Fix bug with omitted values after repeats. SPR 21789 SBL 22-Jan-79
53 0053 1 1-007 - Use 32-bit addresses for externals. JBS 27-JAN-1979
54 0054 1 1-008 - Add support for G, H, DC, GC. Allow lower case exponent
55 0055 1   letters. Have Logical fields fetch second word.
56 0056 1   SBL 21-Mar-79
57 0057 1 1-009 - Fix convert table lookup for G and bigger. SBL 19-Apr-79

```

```
58 0058 1 1-010 - Remove reference to MTH$GFLOTJ. SBL 27-Apr-79
59 0059 1 1-011 - Force integer conversion of repeat constants. SBL 4-May-1979
60 0060 1 1-012 - Remove references to MTH$DFLOTJ, MTH$JIDINT and MTH$SNGL.
61 0061 1 A side effect of this edit is that we now check for overflow
62 0062 1 when converting D to F. JBS 26-JUN-1979
63 0063 1 1-013 - Fix CONVERTTYPE table for datatypes F, FC, DC and GC. SBL 13-Jul-1979
64 0064 1 1-014 - Change to use local CONSBLOCK. Do numeric repeat counts by
65 0065 1 resetting the buffer pointer to the beginning of the numeric
66 0066 1 constant. Make byte variables signed. SBL 18-Jul-1979
67 0067 1 1-015 - If file is not sequential organization, give error
68 0068 1 "mixed file access modes". SBL 3-Oct-1979
69 0069 1 1-016 - Change 1-015 to read not sequential access. SBL 4-Oct-1979
70 0070 1 1-017 - Remove access check, done in FOR$$IO_BEG. SBL 5-Dec-1979
71 0071 1 1-018 - Move BUILTIN ACTUALCOUNT into the routine, in anticipation of the
72 0072 1 next BLISS compiler, which will require it to be there.
73 0073 1 JBS 22-Aug-1980
74 0074 1 1-019 - Change local routine CONVERTTYPE to global routine
75 0075 1 FOR$$CVT TYPE so that NAMELIST can use it. SBL 28-August-1980
76 0076 1 1-020 - Use new F floating input conversion routine, OT$$CVT_i_F.
77 0077 1 14-Apr-1981 JAW
78 0078 1 1-021 - Allow BU datatype to be the same as B. Eliminate differentiation between integer
79 0079 1 and real constant type except where necessary, since the test wasn't guaranteed. SBL 9-June-1981
80 0080 1 1-022 - Signal Access Violations directly, do not change them into I/O Syntax
81 0081 1 Error. DGP 15-Jan-1982
82 0082 1 *** VAX.VMS V3.0
83 0083 1 1-023 - Signal FOR$_LISIO_SYN if first character in numeric field is ')'. This
84 0084 1 error appears not only in the VAX code, but in the OTS's for PDP-11
85 0085 1 FORTRAN IV, FORTRAN IV-PLUS and FORTRAN-77! SBL 7-Sept-1982
86 0086 1 1-024 - JSB to REC-level routines through dispatch table. Signal
87 0087 1 FOR$_INPCONERR instead of FOR$_LISIO_SYN where appropriate. Use
88 0088 1 prologue file. SBL 27-Apr-1983
89 0089 1 1-025 - Don't blank-pad character strings if the constant length was
90 0090 1 greater than 255. SPR 11-57769 S3L 17-June-1983
91 0091 1 --
```

```

93      0092 1  |
94      0093 1  | PROLOGUE FILE:
95      0094 1  |
96      0095 1  |
97      0096 1  | REQUIRE 'RTLIN:FORPROLOG';          FORTRAN definitions
98      0162 1  |
99      0163 1  |
100     0164 1  | TABLE OF CONTENTS:
101     0165 1  |
102     0166 1  |
103     0167 1  | FORWARD ROUTINE
104     0168 1  |   | UDF routines
105     0169 1  |   |
106     0170 1  |   | FOR$$UDF_RLO : JSB_UDFO NOVALUE,
107     0171 1  |   | FOR$$UDF_RL1 : CALL_CCB NOVALUE,
108     0172 1  |   | FOR$$UDF_RL9 : JSB_ODF9 NOVALUE,
109     0173 1  |   |   | routines used by FOR$$UDF_RLO and FOR$$UDF_RL1
110     0174 1  |   |
111     0175 1  |   | GETCONST : CALL_CCB,
112     0176 1  |   | FOR$$CVT TYPE,
113     0177 1  |   | LCL HANDLER,          ! Local handler for conversion routine
114     0178 1  |   | GETFIELD : CALL_CCB,
115     0179 1  |   | SKIPBLANKS : CALL_CCB,
116     0180 1  |   | DELIM : CALL_CCB;
117     0181 1  |   |
118     0182 1  |   |
119     0183 1  |   | MACROS:
120     0184 1  |   |
121     0185 1  |   |
122     0186 1  |   | MACRO
123     0187 1  |   |   | THISCHAR =
124     0188 1  |   |   |   | (IF .CCB[LUB$A_BUF_PTR] GEQA .CCB[LUB$A_BUF_END]
125     0189 1  |   |   |   |   | THEN
126     0190 1  |   |   |   |   |   | -1
127     0191 1  |   |   |   |   | ELSE
128     0192 1  |   |   |   |   |   | .((CCB[LUB$A_BUF_PTR])<0,8>) %,
129     0193 1  |   |   |   |   | NEXTCHAR =
130     0194 1  |   |   |   |   | BEGIN
131     0195 1  |   |   |   |   |   | CCB[LUB$A_BUF_PTR] = .CCB[LUB$A_BUF_PTR] + 1;
132     0196 1  |   |   |   |   |   | THISCHAR
133     0197 1  |   |   |   |   |   | END %;
134     0198 1  |   |   |   |   |
135     0199 1  |   |   |   |   |
136     0200 1  |   |   |   |   | EQUATED SYMBOLS:
137     0201 1  |   |   |   |   |
138     0202 1  |   |   |   |   |
139     0203 1  |   |   |   |   | LITERAL
140     0204 1  |   |   |   |   |   | K_NULL = 0,          ! types of constants which may appear in input record
141     0205 1  |   |   |   |   |   | K_LOG = 1,
142     0206 1  |   |   |   |   |   | K_INT = 2,
143     0207 1  |   |   |   |   |   | K_REAL = 3,
144     0208 1  |   |   |   |   |   | K_COMP = 4,
145     0209 1  |   |   |   |   |   | K_CHAR = 5,
146     0210 1  |   |   |   |   |   | K_TAB = 9;          ! ASCII TAB
147     0211 1  |   |   |   |   |
148     0212 1  |   |   |   |   |
149     0213 1  |   |   |   |   | OWN STORAGE:

```

```
150 0214 1 |  
151 0215 1 | NONE  
152 0216 1 |  
153 0217 1 | EXTERNAL REFERENCES:  
154 0218 1 |  
155 0219 1 |  
156 0220 1 | EXTERNAL ROUTINE  
157 0221 1 | FOR$$GET VM, : Allocate virtual memory  
158 0222 1 | FOR$$FREE VM : NOVALUE, : Deallocate virtual memory  
159 0223 1 | FOR$$SIGNAL STO : NOVALUE, : Signal fatal error  
160 0224 1 | LIB$$SIG_TO_RET, : Convert a signal to a return code  
161 0225 1 | +  
162 0226 1 | conversion routines  
163 0227 1 | -  
164 0228 1 | OT$$CVT_TL_L,  
165 0229 1 | OT$$CVT_TI_L,  
166 0230 1 | OT$$CVT_T_F,  
167 0231 1 | OT$$CVT_T_D,  
168 0232 1 | OT$$CVT_T_G,  
169 0233 1 | OT$$CVT_T_H;  
170 0234 1 |  
171 0235 1 | EXTERNAL  
172 0236 1 | FOR$$AA_REC_PRO: VECTOR, : Self-relative arrays of  
173 0237 1 | FOR$$AA_REC_PR1: VECTOR; : REC-level routine addresses
```

```
175 0238 1 %SBTTL 'FOR$$UDF_RLO'  
176 0239 1 GLOBAL ROUTINE FOR$$UDF_RLO : JSB_UDFO NOVALUE =  
177 0240 1  
178 0241 1 !+  
179 0242 1 FUNCTIONAL DESCRIPTION:  
180 0243 1  
181 0244 1 Perform UDF level read list-directed I/O initialization.  
182 0245 1 Initialize module "own" storage in the ISB.  
183 0246 1 Call record level processor to get first input record.  
184 0247 1  
185 0248 1 CALLING SEQUENCE:  
186 0249 1  
187 0250 1 JSB FOR$$UDF_RLO ()  
188 0251 1  
189 0252 1 FORMAL PARAMETERS:  
190 0253 1  
191 0254 1 NONE  
192 0255 1  
193 0256 1 IMPLICIT INPUTS:  
194 0257 1  
195 0258 1 CCB Pointer to current logical unit block (LUB)  
196 0259 1  
197 0260 1 IMPLICIT OUTPUTS:  
198 0261 1  
199 0262 1 ISB$V_SLASH 0: no slash seen in this record  
200 0263 1 ISB$V_LIS_HEAP 0: no heap storage allocated for string constant  
201 0264 1 ISB$W_LIS_REP 0: no repeat count yet seen  
202 0265 1  
203 0266 1 ROUTINE VALUE:  
204 0267 1 COMPLETION CODES:  
205 0268 1  
206 0269 1 NONE  
207 0270 1  
208 0271 1 SIDE EFFECTS:  
209 0272 1  
210 0273 1 NONE  
211 0274 1  
212 0275 1 --  
213 0276 1  
214 0277 2 BEGIN  
215 0278 2  
216 0279 2 EXTERNAL REGISTER  
217 0280 2 CCB : REF $FOR$CCB_DECL;  
218 0281 2  
219 0282 2 !+  
220 0283 2 Initialize module own storage used between calls to FOR$$UDF_RL1.  
221 0284 2 !-  
222 0285 2  
223 0286 2 CCB [ISB$V_SLASH] = 0;  
224 0287 2 CCB [ISB$W_LIS_REP] = 0;  
225 0288 2 CCB [ISB$V_LIS_HEAP] = 0;  
226 0289 2  
227 0290 2 !+  
228 0291 2 Call record level routine to read the first record.  
229 0292 2 !-  
230 0293 2  
231 0294 2 JSB_RECO (FOR$$AA_REC_PRO + .FOR$$AA_REC_PRO [CCB [ISB$B_STM_TYPE] -
```

FOR\$\$UDF_RL
1-025

FORTTRAN list-directed input, UDF level
FOR\$\$UDF_RLO

I 6
16-Sep-1984 00:47:40
14-Sep-1984 12:32:51

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFRL.B32;1

Page 6
(3)

: 232 0295 2 ISB\$K_FORSTTYLO + 1)),
: 233 0296 1 END;

.TITLE FOR\$\$UDF_RL FORTTRAN list-directed input, UDF le
vel
.IDENT \1-025\
.EXTRN FOR\$\$GET_VM, FOR\$\$FREE_VM
.EXTRN FOR\$\$SIGNAL_STO
.EXTRN LIB\$\$SIG_TO_RET, OT\$\$SCVT_TL_L
.EXTRN OT\$\$SCVT_TI_L, OT\$\$SCVT_TF_L
.EXTRN OT\$\$SCVT_TD, OT\$\$SCVT_TG
.EXTRN OT\$\$SCVT_TH, FOR\$\$AA_REC_PRO
.EXTRN FOR\$\$AA_REC_PRI
.PSECT _FOR\$CODE, NOWRT, SHR, PIC, 2

96 AB 10 8A 0000 FOR\$\$UDF_RLO::
BICB2 #16, -106(CCB)
CLRW -115(CCB)
96 AB 8D AB B4 00004
50 AB 80 8F 8A 00007
50 FF71 CB 9A 0000C
50 00000000G0040 D0 00011
00000000G0040 17 00019
BICB2 #128, -106(CCB)
MOVZBL -143(CCB), R0
MOVL FOR\$\$AA_REC_PRO[R0], R0
JMP FOR\$\$AA_REC_PRO[R0]

: 0286
: 0287
: 0288
: 0295
: 0294
:

: Routine Size: 32 bytes, Routine Base: _FOR\$CODE + 0000

: 234 0297 1


```

236 0298 1 %SBTTL'FOR$$UDF_RL1'
237 0299 1 GLOBAL ROUTINE FOR$$UDF_RL1 (ELEM_TYPE, ELEM_SIZE, ELEM_ADR, FC_FLAG) : CALL_CCB NOVALUE =
238 0300 1
239 0301 1 ++
240 0302 1 FUNCTIONAL DESCRIPTION:
241 0303 1
242 0304 1 Return the next input value to the user I/O list element.
243 0305 1 The value obtained from the input record is converted to
244 0306 1 the type of the list element.
245 0307 1
246 0308 1 CALLING SEQUENCE:
247 0309 1
248 0310 1 CALL FOR$$UDF_RL1 (elem_type.rlu.v, elem_size.rlu.v, elem_adr.wx.r [,fc_flag.rb.v])
249 0311 1
250 0312 1 FORMAL PARAMETERS:
251 0313 1
252 0314 1 ELEM_TYPE.rlu.v Type code of user I/O list element
253 0315 1 ELEM_SIZE.rlu.v Size of user I/O list element
254 0316 1 ELEM_ADR.wx.r Address of user I/O list element,
255 0317 1 x = b,w,l,bu,wu,lu,f,c,fc,dc,gc,g,h or t.
256 0318 1 [FC_FLAG] if present, then:
257 0319 1 0 - real part of COMPLEX type
258 0320 1 1 - imaginary part of COMPLEX type
259 0321 1
260 0322 1 IMPLICIT INPUTS:
261 0323 1
262 0324 1 OTSS$A_CUR_LUB Pointer to current logical unit block (LUB)
263 0325 1
264 0326 1 IMPLICIT OUTPUTS:
265 0327 1
266 0328 1 ISB$W_LIS_HEAP repeat count
267 0329 1 ISB$B_LIS_CTYPE type of constant found
268 0330 1 ISB$A_LIS_STR address of saved repeated string
269 0331 1 ISB$V_HEAP on if heap storage allocated by module
270 0332 1 ISB$V_SLASH on if slash seen (ignore all future calls)
271 0333 1
272 0334 1 ROUTINE VALUE:
273 0335 1 COMPLETION CODES:
274 0336 1
275 0337 1 NONE
276 0338 1
277 0339 1 SIDE EFFECTS:
278 0340 1
279 0341 1 SIGNALs FOR$ LISIO SYN if a bum repeat count or an error
280 0342 1 occurs when converting the constant from external form to
281 0343 1 the type of the list element.
282 0344 1
283 0345 1 --
284 0346 1
285 0347 2 BEGIN
286 0348 2
287 0349 2 EXTERNAL REGISTER
288 0350 2 CCB : REF $FOR$CCB_DECL;
289 0351 2
290 0352 2 MAP
291 0353 2 ELEM_ADR : REF VECTOR;
292 0354 2

```

```
293 0355 2 BUILTIN
294 0356 2 ACTUALCOUNT;
295 0357 2
296 0358 2 LOCAL
297 0359 2 CONSBLOCK : VECTOR [4, LONG],
298 0360 2 CONST_PTR, . Pointer to beginning of constant
299 0361 2 CHARCONS : VECTOR [256, BYTE];
300 0362 2
301 0363 2
302 0364 2 !+
303 0365 2 ! If we're being called to get the second part of a COMPLEX number,
304 0366 2 ! just return since the call for the first part actually got
305 0367 2 ! both parts!
306 0368 2
307 0369 2 IF ACTUALCOUNT () GTR (FC_FLAG - ELEM_TYPE)/%UPVAL
308 0370 2 THEN
309 0371 2
310 0372 2 IF .FC_FLAG THEN RETURN;
311 0373 2
312 0374 2 !+
313 0375 2 ! If a slash has been seen previously, just return as rest of record
314 0376 2 ! is ignored.
315 0377 2
316 0378 2
317 0379 2 IF .CCB [ISBSV_SLASH] THEN RETURN;
318 0380 2
319 0381 2 !+
320 0382 2 ! If no currently active repeat count, find the next constant.
321 0383 2
322 0384 2
323 0385 2 IF .CCB [ISBSW_LIS_REP] EQL 0
324 0386 2 THEN
325 0387 2 BEGIN
326 0388 2
327 0389 2 !+
328 0390 2 ! Find a constant. If a string constant is seen, have GETCONST
329 0391 2 ! store it in stack-local CHARCONS.
330 0392 2
331 0393 2
332 0394 2 SKIPBLANKS ();
333 0395 2 CONSBLOCK [0] = CHARCONS; ! passing address of string area
334 0396 2 CCB [ISBSB_LIS_CTYPE] = GETCONST (CONSBLOCK, 1, .ELEM_TYPE);
335 0397 2
336 0398 2 !+
337 0399 2 ! If the next character after the constant is a star then the
338 0400 2 ! constant is really a repeat count. Make sure the repeat count
339 0401 2 ! is legal and store away in the ISB for future calls.
340 0402 2
341 0403 2
342 0404 2 IF THISCHAR EQL %C'*'
343 0405 2 THEN
344 0406 2 BEGIN
345 0407 2 CCB [LUBSA_BUF_PTR] = .CCB [LUBSA_BUF_PTR] + 1;
346 0408 2 CCB [ISBSW_LIS_REP] = (IF .CCB [ISBSB_LIS_CTYPE] NEQ K_INT OR .CONSBLOCK [0] LEQ 0 THEN
347 0409 2 BEGIN
348 0410 2 CCB [ISBSB_ERR_NO] = FOR$K_LISIO_SYN;
349 0411 2 1
```

```
350 0412 6      END
351 0413 4      ELSE .CONSBLOCK [0]);
352 0414 4
353 0415 4      !+
354 0416 4      ! Now that repeat count is taken care of, get the 'real'
355 0417 4      ! constant!
356 0418 4      !-
357 0419 4
358 0420 4      CONSBLOCK [0] = CHARCONS;
359 0421 4      CONST PTR = .CCB [LUB$A_BUF_PTR]; ! Save address of constant in input
360 0422 4      CCB [ISB$B_LIS_CTYPE] = GETCONST (CONSBLOCK, 1, .ELEM_TYPE);
361 0423 4
362 0424 4      !+
363 0425 4      ! If we just got a string constant (preceded by a repeat count),
364 0426 4      ! then the string must be stored to preserve it between calls
365 0427 4      ! to this routine.
366 0428 4      !-
367 0429 4
368 0430 4      IF .CCB [ISB$B_LIS_CTYPE] EQL K_CHAR
369 0431 4      THEN
370 0432 5          BEGIN
371 0433 5              LOCAL
372 0434 5                  T;
373 0435 5
374 0436 5                  T = FOR$$GET_VM (256);
375 0437 5                  CH$MOVE (255, CHARCONS, .T);
376 0438 5                  CCB [ISB$A_LIS_STR] = .T;
377 0439 5                  CCB [ISB$V_LIS_HEAP] = 1;
378 0440 5                  END
379 0441 5
380 0442 5      END
381 0443 4      ELSE
382 0444 3          CCB [ISB$W_LIS_REP] = 1
383 0445 3
384 0446 3      END
385 0447 3      ELSE
386 0448 2          !+
387 0449 2          ! This is pass 2 or more on a repeat count. If the constant
388 0450 2          ! was not character, call GETCONST to reconvert the value.
389 0451 2          ! Otherwise, put the address of the saved string in
390 0452 2          ! CONSBLOCK [0].
391 0453 2          !-
392 0454 2          IF .CCB [ISB$B_LIS_CTYPE] NEQ K_CHAR
393 0455 2          THEN
394 0456 2              BEGIN
395 0457 2                  CONST PTR = .CCB [LUB$A_BUF_PTR]; ! Save address again
396 0458 2                  CCB [ISB$B_LIS_CTYPE] = GETCONST (CONSBLOCK, 0, .ELEM_TYPE);
397 0459 2                  END
398 0460 2              ELSE
399 0461 2                  CONSBLOCK [0] = .CCB [ISB$A_LIS_STR];
400 0462 2          END
401 0463 2      IF .CCB [ISB$B_LIS_CTYPE] NEQ K_NULL
402 0464 2      THEN
403 0465 2
404 0466 2
405 0467 2
406 0468 2
```

```
407 0469 2 IF NOT FOR$$CVT_TYPE (.CCB [ISB$B_LIS_CTYPE], CONSBLOCK,  
408 0470 2  
409 0471 2 IF ACTUALCOUNT () GTR (FC_FLAG - ELEM_TYPE)/%UPVAL  
410 0472 2 THEN  
411 0473 2  
412 0474 2 SELECTONE .ELEM_TYPE OF  
413 0475 2 SET  
414 0476 2  
415 0477 2 [DSC$K_DTYPE_F] :  
416 0478 2 DSC$K_DTYPE_FC;  
417 0479 2  
418 0480 2 [DSC$K_DTYPE_D] :  
419 0481 2 DSC$K_DTYPE_DC;  
420 0482 2  
421 0483 2 [DSC$K_DTYPE_G] :  
422 0484 2 DSC$K_DTYPE_GC;  
423 0485 2  
424 0486 2 TES  
425 0487 2  
426 0488 2 ELSE  
427 0489 2 .ELEM_TYPE, .ELEM_ADR, .ELEM_SIZE)  
428 0490 2  
429 0491 2 THEN  
430 0492 2 CCB [ISB$B_ERR_NO] = FOR$K_INPCONERR;  
431 0493 2  
432 0494 2 !+ If repeat count goes to zero deallocate heap if there is one  
433 0495 2 ! and skip to next "significant" character.  
434 0496 2 !-  
435 0497 2  
436 0498 2 IF (CCB [ISB$W_LIS_REP] = .CCB [ISB$W_LIS_REP] - 1) EQL 0  
437 0499 2 THEN  
438 0500 2 BEGIN  
439 0501 2  
440 0502 2 LOCAL  
441 0503 2 C; ! Local character storage  
442 0504 2  
443 0505 2 IF .CCB [ISB$V_LIS_HEAP]  
444 0506 2 THEN  
445 0507 2 BEGIN  
446 0508 2 FOR$$FREE VM (256, .CCB [ISB$A_LIS_STR]);  
447 0509 2 CCB [ISB$V_LIS_HEAP] = 0;  
448 0510 2 END;  
449 0511 2  
450 0512 2 !+  
451 0513 2 ! Skip over blanks and tabs until a real character is seen  
452 0514 2 ! or end-of-record is reached. This puts us in a good position  
453 0515 2 ! for the next call.  
454 0516 2 !-  
455 0517 2  
456 0518 2 C = THISCHAR;  
457 0519 2  
458 0520 2 WHILE .C EQL %C' ' OR .C EQL K_TAB DO  
459 0521 2 C = NEXTCHAR;  
460 0522 2  
461 0523 2 IF .C EQL ',' THEN CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1;  
462 0524 2  
463 0525 2 END
```

```

: 464      0526 2      ELSE
: 465      0527 2
: 466      0528 2
: 467      0529 2
: 468      0530 2
: 469      0531 2
: 470      0532 2
: 471      0533 2
: 472      0534 2
: 473      0535 2
: 474      0536 2
: 475      0537 1

```

+

 There is still a repeat count active. If this was a numeric

 value, reset the buffer pointer to point to the beginning of

 the constant for rescanning. If character, the string has

 been saved in the location pointed to by CCB [ISB\$A_LIS_STR].

 -

 IF .CCB [ISB\$B_LIS_TYPE] NEQ K_CHAR THEN CCB [LUB\$A_BUF_PTR] = .CONST_PTR;

 END;

				01FC 00000	.ENTRY	FOR\$\$UDF_RL1, Save R2,R3,R4,R5,R6,R7,R8	: 0299
	58	0000V	CF	9E 00002	MOVAB	GETCONST, R8	
	5E	FEFO	CE	9E 00007	MOVAB	-272(SP), SP	
	03		6C	91 0000C	CMPB	(AP), #3	: 0369
			05	1B 0000F	BLEQU	1\$	
	01	10	AC	E9 00011	BLBC	FC_FLAG, 1\$: 0372
				04 00015	RET		
01	96	AB		E1 00016 1\$:	BBC	#4, -106(CCB), 2\$: 0379
				04 0001B	RET		
			8D	AB B5 0001C 2\$:	TSTW	-115(CCB)	: 0385
				03 13 0001F	BEQL	3\$	
				0090 31 00021	BRW	10\$	
	0000V	CF		00 FB 00024 3\$:	CALLS	#0, SKIPBLANKS	: 0394
	FO	AD		6E 9E 00029	MOVAB	CHARCONS, CONSBLOCK	: 0395
			04	AC DD 0002D	PUSHL	ELEM_TYPE	: 0396
				01 DD 00030	PUSHL	#1	
			FO	AD 9F 00032	PUSHAB	CONSBLOCK	
				03 FB 00035	CALLS	#3, GETCONST	
	8F	AB		50 90 00038	MOVAB	R0, -113(CCB)	
	B4	AB	B0	AB D1 0003C	CMPL	-80(CCB), -76(CCB)	: 0404
				05 1F 00041	BLSSU	4\$	
				01 CE 00043	MNEGL	#1, R0	
				04 11 00046	BRB	5\$	
			B0	BB 9A 00048 4\$:	MOVZBL	@-80(CCB), R0	
				50 D1 0004C 5\$:	CMPL	R0, #42	
				5D 12 0004F	BNEQ	9\$	
			B0	AB D6 00051	INCL	-80(CCB)	: 0407
				8F AB 91 00054	CMPB	-113(CCB), #2	: 0408
				05 12 00058	BNEQ	6\$	
			FO	AD D5 0005A	TSTL	CONSBLOCK	
				0A 14 0005D	BGTR	7\$	
	FF70	CB		3B 90 0005F 6\$:	MOVAB	#59, -144(CCB)	: 0410
				01 D0 00064	MOVL	#1, R0	: 0409
				04 11 00067	BRB	8\$	
			FO	AD D0 00069 7\$:	MOVL	CONSBLOCK, R0	: 0413
				50 B0 0006D 8\$:	MOVW	R0, -115(CCB)	: 0408
	8D	AB		6E 9E 00071	MOVAB	CHARCONS, CONSBLOCK	: 0420
	FO	AD		B0 AB D0 00075	MOVL	-80(CCB), CONST_PTR	: 0421
				04 AC DD 00079	PUSHL	ELEM_TYPE	: 0422
				01 DD 0007C	PUSHL	#1	

			FO	AD	9F	0007E	PUSHAB	CONSBLOCK	
				03	FB	00081	CALLS	#3, GETCONST	
	8F	68		50	90	00084	MOVB	RO, -113(CCB)	
		05	8F	AB	91	00088	CMPB	-113(CCB), #5	0430
				46	12	0008C	BNEQ	12\$	
		7E	0100	8F	3C	0008E	MOVZWL	#256, -(SP)	0437
00000000G		00		01	FB	00093	CALLS	#1, FOR\$\$GET_VM	
		56		50	D0	0009A	MOVL	RO, T	
66		6E	00FF	8F	28	0009D	MOV3	#255, CHARCONS, (T)	0438
	84	AB		56	D0	000A3	MOVL	T, -124(CCB)	0439
	96	AB	80	8F	88	000A7	BISB2	#128, -106(CCB)	0440
				26	11	000AC	BRB	12\$	0430
	8D	AB		01	B0	000AE	MOVW	#1, -115(CCB)	0445
				20	11	000B2	BRB	12\$	0404
		05	8F	AB	91	000B4	CMPB	-113(CCB), #5	0457
				15	13	000B8	BEQL	11\$	
		57	B0	AB	D0	000BA	MOVL	-80(CCB), CONST_PTR	0460
			04	AC	DD	000BE	PUSHL	ELEM_TYPE	0461
				7E	D4	000C1	CLRL	-(SP)	
			FO	AD	9F	000C3	PUSHAB	CONSBLOCK	
		68		03	FB	000C6	CALLS	#3, GETCONST	
	8F	AB		50	90	000C9	MOVB	RO, -113(CCB)	
				05	11	000CD	BRB	12\$	0457
	FO	AD	84	AB	D0	000CF	MOVL	-124(CCB), CONSBLOCK	0464
				8F	AB	95	TSTB	-113(CCB)	0466
				4C	13	000D7	BEQL	19\$	
			08	AC	DD	000D9	PUSHL	ELEM_SIZE	0488
			0C	AC	DD	000DC	PUSHL	ELEM_ADR	
		03		6C	91	000DF	CMPB	(AP), #3	0471
				29	1B	000E2	BLEQU	17\$	
		50	04	AC	D0	000E4	MOVL	ELEM_TYPE, RO	0474
	0A			50	D1	000E8	CPL	RO, #10	0477
				05	12	000EB	BNEQ	13\$	
		50		0C	D0	000ED	MOVL	#12, RO	
				17	11	000F0	BRB	16\$	
		0B		50	D1	000F2	CPL	RO, #11	0480
				05	12	000F5	BNEQ	14\$	
		50		0D	D0	000F7	MOVL	#13, RO	
				0D	11	000FA	BRB	16\$	
		1B		50	D1	000FC	CPL	RO, #27	0483
				05	13	000FF	BEQL	15\$	
		50		01	CE	00101	MNEGL	#1, RO	
				03	11	00104	BRB	16\$	
		50		1D	D0	00106	MOVL	#29, RO	0474
				50	DD	00109	PUSHL	RO	0474
				03	11	0010B	BRB	18\$	
			04	AC	DD	0010D	PUSHL	ELEM_TYPE	0488
			FO	AD	9F	00110	PUSHAB	CONSBLOCK	0469
				8F	AB	9A	MOVZBL	-113(CCB), -(SP)	
	0000V	7E		05	FB	00117	CALLS	#5, FOR\$\$CVT_TYPE	
		CF		50	E8	0011C	BLBS	RO, 19\$	
		06		8F	90	0011F	MOVB	#64, -144(CCB)	0491
	FF70	CB	40	8F	90	0011F	MOVB	#64, -144(CCB)	0491
		50	8D	AB	3C	00125	MOVZWL	-115(CCB), RO	0498
				50	D7	00129	DECL	RO	
		8D		50	B0	0012B	MOVW	RO, -115(CCB)	
				50	D5	0012F	TSTL	RO	
				41	12	00131	BNEQ	25\$	

		96	AB	95	00133		TSTB	-106(CCB)	:	0505
			27	18	00136		BGEQ	23\$:	
		84	AB	DD	00138		PUSHL	-124(CCB)	:	0508
			8F	3C	0013B		MOVZWL	#256, -(SP)	:	
00000000G	7E	0100	02	FB	00140		CALLS	#2, FOR\$\$FREE VM	:	
	96	AB	80	8F	8A	00147	BICB2	#128, -106(CCB)	:	0509
			11	11	0014C		BRB	23\$:	0518
	50		B0	BB	9A	0014E	20\$: MOVZBL	@-80(CCB), C	:	
	20			50	D1	00152	21\$: CMPL	C, #32	:	0520
				05	13	00155	BEQL	22\$:	
	09			50	D1	00157	CMPL	C, #9	:	
				0F	12	0015A	BNEQ	24\$:	
		B0	AB	D6	0015C	22\$: INCL	-80(CCB)	:	0521	
B4	AB	B0	AB	D1	0015F	23\$: CMPL	-80(CCB), -76(CCB)	:		
				E8	1F	00164	BLSSU	20\$:	
	50			01	CE	00166	MNEGL	#1, C	:	
				E7	11	00169	BRB	21\$:	
	2C			50	D1	0016B	24\$: CMPL	C, #44	:	0523
				0E	12	0016E	BNEQ	26\$:	
		B0	AB	D6	00170		INCL	-80(CCB)	:	
				04	00173		RET		:	0498
	05		8F	AB	91	00174	25\$: CMPB	-113(CCB), #5	:	0535
				04	13	00178	BEQL	26\$:	
B0	AB			57	D0	0017A	MOVL	CONST_PTR, -80(CCB)	:	
				04	0017E	26\$: RET		:	0537	

; Routine Size: 383 bytes, Routine Base: _FOR\$CODE + 0020

; 476 0538 1

```

: 478 0539 1 %SBTTL'FOR$$UDF_RL9'
: 479 0540 1 GLOBAL ROUTINE FOR$$UDF_RL9 : JSB_UDF9 NOVALLE =
: 480 0541 1
: 481 0542 1 +-
: 482 0543 1 FUNCTIONAL DESCRIPTION:
: 483 0544 1
: 484 0545 1 List directed input UDF termination:
: 485 0546 1 If any heap storage was allocated by RL1, deallocate it.
: 486 0547 1
: 487 0548 1 CALLING SEQUENCE:
: 488 0549 1
: 489 0550 1 JSB FOR$$UDF_RL9 ( )
: 490 0551 1
: 491 0552 1 FORMAL PARAMETERS:
: 492 0553 1
: 493 0554 1 NONE
: 494 0555 1
: 495 0556 1 IMPLICIT INPUTS:
: 496 0557 1
: 497 0558 1 CCB Adr. of LUB/ISB/RAB
: 498 0559 1 CCB[ISB$V_LIS_HEAP] 1 if storage currently allocated
: 499 0560 1 CCB[ISB$A_LIS_STR] address of allocated storage
: 500 0561 1
: 501 0562 1 IMPLICIT OUTPUTS:
: 502 0563 1
: 503 0564 1 CCB[ISB$V_LIS_HEAP] 0
: 504 0565 1
: 505 0566 1 ROUTINE VALUE:
: 506 0567 1 COMPLETION CODES:
: 507 0568 1
: 508 0569 1 NONE
: 509 0570 1
: 510 0571 1 SIDE EFFECTS:
: 511 0572 1
: 512 0573 1 NONE
: 513 0574 1
: 514 0575 1 --
: 515 0576 1
: 516 0577 2 BEGIN
: 517 0578 2
: 518 0579 2 EXTERNAL REGISTER
: 519 0580 2 CCB : REF $FOR$CCB_DECL;
: 520 0581 2
: 521 0582 2 IF .CCB [ISB$V_LIS_HEAP]
: 522 0583 2 THEN
: 523 0584 3 BEGIN
: 524 0585 3 FOR$$FREE VM (256, .CCB [ISB$A_LIS_STR]);
: 525 0586 3 CCB [ISB$V_LIS_HEAP] = 0;
: 526 0587 3 END;
: 527 0588 2
: 528 0589 1 END;

```


FOR\$\$UDF_RL
1-025

FORTTRAN list-directed input, UDF level
FOR\$\$UDF_RL9

E 7
16-Sep-1984 00:47:40
14-Sep-1984 12:32:51

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFRL.B32;1

Page 15
(5)

FO
1-

			14	18	00003	TSTB	-106(CCB)	:	0582
		84	AB	DD	00005	BGEQ	1\$:	
		0100	8F	3C	00008	PUSHL	-124(CCB)	:	0585
00000000G	7E		02	FB	0000D	MOVZWL	#256, -(SP)	:	
	00		8F	8A	00014	CALLS	#2, FOR\$\$FREE VM	:	
	96	AB	80	05	00019	BICB2	#128, -106(CCB)	:	0586
						RSB		:	0589

: Routine Size: 26 bytes, Routine Base: _FOR\$CODE + 019F

: 529 0590 1

```
531 0591 1 %SBTTL'FOR$$CVT_TYPE'  
532 0592 1 GLOBAL ROUTINE FOR$$CVT_TYPE (IN_TYPE, IN_BLOCK, OUT_TYPE, OUT_BLOCK, OUT_SIZE) =  
533 0593 1  
534 0594 1 ++  
535 0595 1 Functional description:  
536 0596 1  
537 0597 1 Convert the constant recovered from the input record to the  
538 0598 1 type the user requested. If the input and output types  
539 0599 1 are both string constant, copy the string to the users area.  
540 0600 1  
541 0601 1 Formal parameters:  
542 0602 1  
543 0603 1 IN_TYPE.rx.v (L*4, I*4, REAL, CMPLX, CHAR)  
544 0604 1 IN_BLOCK.rl.r address of the input constant  
545 0605 1 OR if the input is a char constant, then  
546 0606 1 the address of a pointer to the char constant.  
547 0607 1 OUT_TYPE.rl.v (BU, WU, LU, B, W, L, F, D, FC, DC, GC, G, H or I)  
548 0608 1 OUT_ADR.wy.r address of output area in user program  
549 0609 1 OUT_SIZE.rl.v size of users output area (used for strings only)  
550 0610 1  
551 0611 1 Returned value:  
552 0612 1 returns success(1) or failure(0) when conversion error occurs.  
553 0613 1  
554 0614 1 --  
555 0615 1  
556 0616 2 BEGIN  
557 0617 2  
558 0618 2 MACRO  
559 M 0619 2 B_0 =  
560 0620 2 0,0,8,1 % ! first byte (signed)  
561 M 0621 2 W_0 =  
562 0622 2 0,0,16,1 % ! first word (sign extend)  
563 M 0623 2 W_1 =  
564 0624 2 0,16,16,0 % ! second word  
565 M 0625 2 L_0 =  
566 0626 2 0,0,32,0 % ! first longword  
567 M 0627 2 L_1 =  
568 0628 2 4,0,32,0 % ! second longword  
569 M 0629 2 L_2 =  
570 0630 2 8,0,32,0 % ! third longword  
571 M 0631 2 L_3 =  
572 0632 2 12,0,32,0 % ! fourth longword  
573 0633 2 ! fields used to access flag bits in FLAG  
574 0634 2 !  
575 M 0635 2 LOAD_FIRST_WORD =  
576 0636 2 0,0,T,0 % ! 000001  
577 M 0637 2 LOAD_SEC_WORD =  
578 0638 2 0,1,T,0 % ! 000002  
579 M 0639 2 LOAD_SEC_LONG =  
580 0640 2 0,2,T,0 % ! 000004  
581 M 0641 2 CONV_J_TO_D =  
582 0642 2 0,3,T,0 % ! 000010  
583 M 0643 2 CONV_D_TO_J =  
584 0644 2 0,4,T,0 % ! 000020  
585 M 0645 2 CONV_D_TO_F =  
586 0646 2 0,5,T,0 % ! 000040  
587 M 0647 2 CONV_J_TO_I =
```

```

588      0,6,1,0 %      : 000100
589      M 0649      222  CONV_J_TO_B =
590      0650      222  0,7,1,0 %      : 000200
591      M 0651      222  STOR_FIRST_BYTE =
592      0652      222  0,8,1,0 %      : 000400
593      M 0653      222  STOR_FIRST_WORD =
594      0654      222  0,9,1,0 %      : 001000
595      M 0655      222  STOR_SEC_WORD =
596      0656      222  0,10,1,0 %      : 002000
597      M 0657      222  STOR_SEC_LONG =
598      0658      222  0,11,1,0 %      : 004000
599      M 0659      222  LOAD_SEC_QUAD =
600      0660      222  0,12,1,0 %      : 010000
601      M 0661      222  STOR_SEC_QUAD =
602      0662      222  0,13,1,0 %      : 020000
603      M 0663      222  CONV_L_TO_FDGH =
604      0664      222  0,14,1,0 %      : 040000
605      0665
606      0666
607      0667      222  LOCAL
608      0668      222  FLAGS : BLOCK [1],
609      0669      222  T : BLOCK [16, BYTE];      ! Local temp storage for intermediate results
610      0670
611      0671      222  MAP
612      0672      222  IN_BLOCK : REF BLOCK [16, BYTE];      ! Contains input value
613      0673      222  OUT_BLOCK : REF BLOCK [16, BYTE];      ! Contains output value
614      0674
615      0675      222  BIND
616      0676      222  FLAG_TAB = UPLIT WORD
617      0677      222  :
618      0678      222  INPUT DATA TYPE
619      0679      222  LOG          INT          REAL          CMPLX
620      0680      222  (
621      0681      222  (
622      0682      222  X0'603'      X0'603'      X0'627'      X0'401'      ! BU (same as B)
623      0683      222  X0'1001'     X0'1103'     X0'1127'     X0'1001'     ! WU
624      0684      222  X0'3003'     X0'3003'     X0'3027'     X0'3003'     ! LU
625      0685      222  0,          0,          0,          0,          ! QU (not used)
626      0686      222  X0'603'      X0'603'      X0'627'      X0'401'     ! B
627      0687      222  X0'1001'     X0'1103'     X0'1127'     X0'1001'     ! W
628      0688      222  X0'3003'     X0'3003'     X0'3027'     X0'3003'     ! L
629      0689      222  0,          0,          0,          0,          ! Q (not used)
630      0690      222  X0'43003'     X0'3003'     X0'3003'     X0'3003'     ! F
631      0691      222  X0'47003'     X0'7007'     X0'7007'     X0'7007'     ! D
632      0692      222  X0'47003'     X0'7003'     X0'7003'     X0'7007'     ! FC
633      0693      222  X0'67003'     X0'27007'     X0'27007'     X0'37007'     ! DC
634      0694      222  X0'47003'     X0'7007'     X0'7007'     X0'7007'     ! G
635      0695      222  X0'67003'     X0'37007'     X0'37007'     X0'37007'     ! H
636      0696      222  X0'67003'     X0'27007'     X0'27007'     X0'37007'     ! GC
637      0697
638      0698      222  : VECTOR [, WORD];
639      0699
640      0700      222  ENABLE
641      0701      222  LCL_HANDLER();
642      0702
643      0703      222  IF .IN_TYPE EQL K_CHAR AND .OUT_TYPE EQL DSC$K_DTYPE_T
644      0704      222  THEN

```

```
645 0705 3 BEGIN
646 0706 3 CH$COPY (255, ..IN_BLOCK, %C' ', .OUT_SIZE, .OUT_BLOCK);
647 0707 3 RETURN 1;
648 0708 2 END;
649 0709 2
650 0710 2 IF .IN_TYPE EQL K_CHAR OR .OUT_TYPE EQL DSC$K_DTYPE_T THEN RETURN 0; ! type mis-match!
651 0711 2
652 0712 4 FLAGS [L_0] = .FLAG_TAB [(OUT_TYPE - DSC$K_DTYPE_BU - (IF .OUT_TYPE GEQU DSC$K_DTYPE_G THEN
653 0713 2 DSC$K_DTYPE_G - DSC$K_DTYPE_DC - 1 ELSE 0))*4 + (.IN_TYPE - K_LOG)];
654 0714 2
655 0715 2 !+
656 0716 2 ! Zero the third and fourth longwords of T so that storing short values
657 0717 2 ! into longer ones works.
658 0718 2 !-
659 0719 2
660 0720 2 T [L_2] = 0;
661 0721 2 T [L_3] = 0;
662 0722 2
663 0723 2 IF .FLAGS [LOAD_FIRST_WORD] ! load first word and sign extend
664 0724 2 THEN
665 0725 2 T [L_0] = .IN_BLOCK [W_0];
666 0726 2
667 0727 2 IF .FLAGS [LOAD_SEC_WORD] ! load second word
668 0728 2 THEN
669 0729 2 T [W_1] = .IN_BLOCK [W_1];
670 0730 2
671 0731 2 IF .FLAGS [LOAD_SEC_LONG] ! load third and fourth words
672 0732 2 THEN
673 0733 2 T [L_1] = .IN_BLOCK [L_1];
674 0734 2
675 0735 2 IF .FLAGS [LOAD_SEC_QUAD] ! load second quadword
676 0736 2 THEN
677 0737 3 BEGIN
678 0738 3 T [L_2] = .IN_BLOCK [L_2];
679 0739 3 T [L_3] = .IN_BLOCK [L_3];
680 0740 2 END;
681 0741 2
682 0742 2 IF .FLAGS [CONV_J_TO_D] ! convert J to D
683 0743 2 THEN
684 0744 3 BEGIN
685 0745 3 BUILTIN
686 0746 3 CVTLD;
687 0747 3
688 0748 3 CVTLD (T [L_0], T [L_0]);
689 0749 3 END;
690 0750 2
691 0751 2
692 0752 2 IF .FLAGS [CONV_L_TO_FDGH] ! Convert Logical to floating
693 0753 2 THEN
694 0754 3 BEGIN
695 0755 3
696 0756 3 !+
697 0757 3 ! If the logical value is true, set the floating value to -1.
698 0758 3 ! Otherwise it is already zero.
699 0759 3 !-
700 0760 3
701 0761 3 IF .T [L_0]
```

```

: 702      0762      3      THEN
: 703      0763      3          T [L_0] =
: 704      0764      4          BEGIN
: 705      0765      4
: 706      0766      4          SELECT ONE .OUT_TYPE OF
: 707      0767      4          SET
: 708      0768      4
: 709      0769      4          [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_FC, DSC$K_DTYPE_DC] :
: 710      0770      4          %X'COB0';
: 711      0771      4
: 712      0772      4          [DSC$K_DTYPE_G, DSC$K_DTYPE_GC] :
: 713      0773      4          %X'CO10';
: 714      0774      4
: 715      0775      4          [DSC$K_DTYPE_H] :
: 716      0776      4          %X'CO01';
: 717      0777      4          TES
: 718      0778      4
: 719      0779      3          END;
: 720      0780      3
: 721      0781      3          T [L_1] = 0;
: 722      0782      3          T [L_2] = 0;
: 723      0783      3          T [L_3] = 0;
: 724      0784      2          END;
: 725      0785      2
: 726      0786      2      IF .FLAGS [CONV_D_TO_J]          ! convert D to J
: 727      0787      2      THEN
: 728      0788      3          BEGIN
: 729      0789      3          BUILTIN
: 730      0790      3          CVTDL;
: 731      0791      3
: 732      0792      3          IF ( NOT CVTDL (T [L_0], T [L_0])) THEN RETURN 0;
: 733      0793      3
: 734      0794      3          T [L_1] = 0;
: 735      0795      3          END;
: 736      0796      2
: 737      0797      2
: 738      0798      2      IF .FLAGS [CONV_D_TO_F]          ! convert D to F (round)
: 739      0799      2      THEN
: 740      0800      3          BEGIN
: 741      0801      3          BUILTIN
: 742      0802      3          CVTDF;
: 743      0803      3
: 744      0804      3          IF ( NOT CVTDF (T [L_0], T [L_0])) THEN RETURN 0;
: 745      0805      3
: 746      0806      3          T [L_1] = 0;
: 747      0807      3          END;
: 748      0808      2
: 749      0809      2
: 750      0810      2      IF .FLAGS [CONV_J_TO_I]          ! convert longword to word (signed)
: 751      0811      2      THEN
: 752      0812      2
: 753      0813      2          IF .T [0, 15, 1, 1] NEQ .T [0, 16, 16, 1] THEN RETURN 0;
: 754      0814      2
: 755      0815      2      IF .FLAGS [CONV_J_TO_B]          ! convert longword to byte (signed)
: 756      0816      2      THEN
: 757      0817      2
: 758      0818      2          IF .T [0, 7, 1, 1] NEQ .T [0, 8, 24, 1] THEN RETURN 0;
```

```

: 759      0819      2
: 760      0820      2      IF .FLAGS [STOR_FIRST_BYTE]      ! store one byte
: 761      0821      2      THEN
: 762      0822      2      OUT_BLOCK [B_0] = .T [L_0];
: 763      0823      2
: 764      0824      2      IF .FLAGS [STOR_FIRST_WORD]      ! store one word
: 765      0825      2      THEN
: 766      0826      2      OUT_BLOCK [W_0] = .T [W_0];
: 767      0827      2
: 768      0828      2      IF .FLAGS [STOR_SEC_WORD]      ! store second word
: 769      0829      2      THEN
: 770      0830      2      OUT_BLOCK [W_1] = .T [W_1];
: 771      0831      2
: 772      0832      2      IF .FLAGS [STOR_SEC_LONG]      ! store third and fourth words
: 773      0833      2      THEN
: 774      0834      2      OUT_BLOCK [L_1] = .T [L_1];
: 775      0835      2
: 776      0836      2      IF .FLAGS [STOR_SEC_QUAD]      ! store second quadword
: 777      0837      2      THEN
: 778      0838      3      BEGIN
: 779      0839      3      OUT_BLOCK [L_2] = .T [L_2];
: 780      0840      3      OUT_BLOCK [L_3] = .T [L_3];
: 781      0841      2      END;
: 782      0842      2
: 783      0843      2      RETURN 1;
: 784      0844      1      END;

```

0603	0603	0201	0257	0243	0201	0101	0197	0183	0183	001B9	P.AAA:	.BLKB	1	387	387	407	257	513	579	599	513	-	:	
0101	0197	0183	0183	0000	0000	0000	0000	0603	0617	001BA	.WORD			1539	1539	1559	1539	0	0	0	0	387	-	:
0000	0000	0603	0617	0603	0603	0201	0257	0243	0201	001CE				387	407	257	513	579	599	513	1539	-	:	
0E07	0E07	0E07	4E03	0603	0603	0603	4603	0000	0000	001F6				1539	1559	1539	0	0	0	0	17923	-	:	
0E07	4E03	3E07	2E07	2E07	6E03	0E07	0E03	0E03	4E03	0020A				1539	1539	1539	19971	3591	3591			-	:	
3E07	2E07	2E07	6E03	3E07	3E07	3E07	6E03	0E07	0E07	0021E				3591	19971	3587	3587	3591	28163			-	:	
														11783	11783	15879	19971	3591	3591			-	:	
														3591	28163	15879	15879	15879	28163			-	:	
														11783	11783	15879						-	:	

FLAG_TAB= P.AAA

				007C	00000	.ENTRY	FOR\$\$CVT_TYPE, Save R2,R3,R4,R5,R6	:	0592
5E			10	C2	00002	SUBL2	#16, SP	:	
6D	013C		CF	DE	00005	MOVAL	28\$, (FP)	:	0616
			56	D4	0000A	CLRL	R6	:	0703
05	04		AC	D1	0000C	CMPL	IN_TYPE, #5	:	
			19	12	00010	BNEQ	1\$:	
			56	D6	00012	INCL	R6	:	
0E	0C		AC	D1	00014	CMPL	OUT_TYPE, #14	:	
			11	12	00018	BNEQ	1\$:	
14	AC		BC	D0	0001A	MOVL	@IN_BLOCK, R0	:	0706
		20	00FF	8F	2C	MOVCS	#255, (R0), #32, OUT_SIZE, @OUT_BLOCK	:	
			10	BC	00026			:	
			0113	31	00028	BRW	26\$:	0707
03			56	E9	0002B	BLBC	R6, 3\$:	0710

			0111	31	0002E	2\$:	BRW	27\$		
	0E	OC	AC	D1	00031	3\$:	CMPL	OUT_TYPE, #14		
			F7	13	00035		BEQL	2\$		
	51	OC	AC	D0	00037		MOVL	OUT_TYPE, R1		0712
	1B		51	D1	0003B		CMPL	R1, #27		
			05	1F	0003E		BLSSU	4\$		
	50		0D	D0	00040		MOVL	#13, R0		0713
			02	11	00043		BRB	5\$		
		50	50	D4	00045	4\$:	CLRL	R0		0712
	51		50	C3	00047	5\$:	SUBL3	R0, R1, R0		
	50	04	BC40	DE	0004B		MOVAL	@IN_TYPE[R0], R0		0713
	52	FF21	CF40	3C	00050		MOVZWL	FLAG_TAB-18[R0], FLAGS		0712
		08	AE	7C	00056		CLRQ	T+8		0720
	04		52	E9	00059		BLBC	FLAGS, 6\$		0723
	6E	08	BC	32	0005C		CVTDL	@IN_BLOCK, T		0725
	52		01	E1	00060	6\$:	BBC	#1, FLAGS, 7\$		0727
50	10	0A	10	EF	00064		EXTZV	#16, #16, @IN_BLOCK, R0		0729
	AE	08	50	B0	0006A		MOVW	R0, T+2		
	52	09	02	E1	0006E	7\$:	BBC	#2, FLAGS, 8\$		0731
	50		08	AC	00072		MOVL	IN_BLOCK, R0		0733
	AE	09	04	A0	00076		MOVL	4(R0), T+4		
	52		0C	E1	0007B	8\$:	BBC	#12, FLAGS, 9\$		0735
	50		08	AC	0007F		MOVL	IN_BLOCK, R0		0738
	AE	03	08	A0	00083		MOVQ	8(R0), T+8		
	52		03	E1	00088	9\$:	BBC	#3, FLAGS, 10\$		0742
	6E		6E	6E	0008C		CVTDL	T, T		0749
	52	3D	0E	E1	0008F	10\$:	BBC	#14, FLAGS, 17\$		0752
	34		6E	E9	00093		BLBC	T, 16\$		0761
	0A		51	D1	00096		CMPL	R1, #10		0769
			0C	19	00099		BLSS	11\$		
	0D		51	D1	0009B		CMPL	R1, #13		
			07	14	0009E		BGTR	11\$		
	50	C080	8F	3C	000A0		MOVZWL	#49280, R0		
			20	11	000A5		BRB	15\$		
	1B		51	D1	000A7	11\$:	CMPL	R1, #27		0772
			05	13	000AA		BEQL	12\$		
	1D		51	D1	000AC		CMPL	R1, #29		
			07	12	000AF		BNEQ	13\$		
	50	C010	8F	3C	000B1	12\$:	MOVZWL	#49168, R0		
			0F	11	000B6		BRB	15\$		
	1C		51	D1	000B8	13\$:	CMPL	R1, #28		0775
			05	13	000BB		BEQL	14\$		
	50		01	CE	000BD		MNEGL	#1, R0		
			05	11	000C0		BRB	15\$		
	50	C001	8F	3C	000C2	14\$:	MOVZWL	#49153, R0		
	6E		50	D0	000C7	15\$:	MOVL	R0, T		0764
		04	AE	7C	000CA	16\$:	CLRQ	T+4		0781
		0C	AE	D4	000CD		CLRL	T+12		0783
	08		52	04	000D0	17\$:	BBC	#4, FLAGS, 18\$		0786
	6E		6E	6A	000D4		CVTDL	T, T		0793
			69	1D	000D7		BVS	27\$		
		04	AE	D4	000D9		CLRL	T+4		0795
	08		52	05	000DC	18\$:	BBC	#5, FLAGS, 19\$		0798
	6E		6E	76	000E0		CVTDF	T, T		0805
			5D	1D	000E3		BVS	27\$		
		04	AE	D4	000E5		CLRL	T+4		0807
	OC		52	06	000E8	19\$:	BBC	#6, FLAGS, 20\$		0810

50	01	AE	50	02	AE	32	000EC		CVTWL	T+2, R0	0813
			01		07	EC	000FO		CMPV	#7, #1, T+1, R0	
					4A	12	000F6		BNEQ	27\$	
					52	95	000F8	20\$:	TSTB	FLAGS	0815
					0D	18	000FA		BGEQ	21\$	
50	01	AE	18		00	EE	000FC		EXTV	#0, #24, T+1, R0	0818
50		6E	C1		07	EC	00102		CMPV	#7, #1, T, R0	
					39	12	00107		BNEQ	27\$	
		04	52		08	E1	00109	21\$:	BBC	#8, FLAGS, 22\$	0820
			10	BC	6E	90	0010D		MOVB	T, @OUT_BLOCK	0822
		04	52		09	E1	00111	22\$:	BBC	#9, FLAGS, 23\$	0824
			10	BC	6E	B0	00115		MOVW	T, @OUT_BLOCK	0826
		07	52		0A	E1	00119	23\$:	BBC	#10, FLAGS, 24\$	0828
10	BC		10		02	AE	F0	0011D	INSV	T+2, #16, #16, @OUT_BLOCK	083
		09	52		0B	E1	00124	24\$:	BBC	#11, FLAGS, 25\$	0832
			50		10	AC	D0	00128	MOVL	OUT_BLOCK, R0	0834
		09	A0	04	AE	D0	0012C		MOVL	T+4, 4(R0)	
			52		0D	E1	00131	25\$:	BBC	#13, FLAGS, 26\$	0836
			50		10	AC	D0	00135	MOVL	OUT_BLOCK, R0	0839
		09	A0	08	AE	7D	00139		MOVQ	T+8, 8(R0)	
			50		01	D0	0013E	26\$:	MOVL	#1, R0	0843
						04	00141		RET		
					50	D4	00142	27\$:	CLRL	R0	0844
						04	00144		RET		
						0000	00145	28\$:	.WORD	Save nothing	0616
					7E	D4	00147		CLRL	-(SP)	
					5E	DD	00149		PUSHL	SP	
			7E	04	AC	7D	0014B		MOVQ	4(AP), -(SP)	
		0000V	CF		03	FB	0014F		CALLS	#3, LCL_HANDLER	
					04	00154			RET		

; Routine Size: 341 bytes, Routine Base: _FOR\$CODE + 0232


```
786 0845 1 %SBTTL 'GETCONST'
787 0846 1 ROUTINE GETCONST (CONSBLOCK, STRINGFLAG, ELEM_TYPE) : CALL_CCB =
788 0847 1
789 0848 1 ++
790 0849 1 FUNCTIONAL DESCRIPTION:
791 0850 1
792 0851 1 Obtain a value from the external record using the format conversion
793 0852 1 routines. The conversion chosen is dependent on the contents of the
794 0853 1 field of the record:
795 0854 1 LOGICAL if the first char is 'T', 't', 'F', 'f' or the first
796 0855 1 character is '.' and the second is any of the above.
797 0856 1 COMPLEX if the first char is '(';
798 0857 1 CHAR if first char is ';
799 0858 1 FLOATING otherwise.
800 0859 1
801 0860 1 A special case is made if the next character after the value
802 0861 1 is '*', in which case it is a repeat count and is always
803 0862 1 converted to integer.
804 0863 1
805 0864 1 FORMAL PARAMETERS:
806 0865 1
807 0866 1 CONSBLOCK.mb.r Two longword block in which to store
808 0867 1 the constant found.
809 0868 1 STRINGFLAG 0 if caller wishes not to have strings
810 0869 1 returned to him.
811 0870 1 1 if caller wants string returned:
812 0871 1 CONSBLOCK[0] contains the address of
813 0872 1 the 255 byte area to store the string.
814 0873 1 ELEM_TYPE The datatype of the destination.
815 0874 1
816 0875 1 IMPLICIT INPUTS:
817 0876 1
818 0877 1
819 0878 1 IMPLICIT OUTPUTS:
820 0879 1
821 0880 1 If a string constant is seen and STRINGFLAG is one, the
822 0881 1 string will be stored starting at the address specified in
823 0882 1 CONSBLOCK[0]. The string will always be 255 bytes long (blank
824 0883 1 padded).
825 0884 1
826 0885 1 ROUTINE VALUE:
827 0886 1
828 0887 1 The type of the constant seen is returned (as a small number)
829 0888 1 as the routine value.
830 0889 1
831 0890 1 COMPLETION CODES:
832 0891 1
833 0892 1 NONE
834 0893 1
835 0894 1 SIDE EFFECTS:
836 0895 1
837 0896 1 SIGNALS FOR$LISIO_SYN if a conversion error occurs.
838 0897 1
839 0898 1 --
840 0899 1
841 0900 2 BEGIN
842 0901 2
```

```
843 0902 2 EXTERNAL REGISTER
844 0903 CCB : REF $FOR$CCB_DECL;
845 0904
846 0905 MAP
847 0906 CONSBLOCK : REF VECTOR [4];
848 0907
849 0908 LOCAL
850 0909 CTYPE : BYTE ! local type of constant seen
851 0910 DSC : BLOCK [8, BYTE];
852 0911
853 0912
854 0913 + The first character indicates what type of constant this is.
855 0914 - Perform the appropriate action.
856 0915
857 0916
858 0917 SELECTONE THISCHAR OF
859 0918 SET
860 0919
861 0920 +
862 0921 - Complex constant
863 0922
864 0923
865 0924 [%C '('] :
866 0925 BEGIN
867 0926
868 0927 LOCAL
869 0928 L_CONSBLOCK : VECTOR [4],
870 0929 B_CTYPE : BYTE,
871 0930 B_ERR_FLAG : BYTE;
872 0931
873 0932 +
874 0933 - Set error flag to 0, skip over the '('.
875 0934
876 0935
877 0936 B_ERR_FLAG = 0;
878 0937 CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1;
879 0938
880 0939 +
881 0940 - Skip over any leading blanks and commas. If a comma is seen
882 0941 this is an error.
883 0942
884 0943
885 0944 IF DELIM ( ) THEN B_ERR_FLAG = .B_ERR_FLAG + 1;
886 0945
887 0946 +
888 0947 - Get the next constant from the input record. Strings
889 0948 are not allowed at this time!
890 0949
891 0950
892 0951 B_CTYPE = GETCONST (L_CONSBLOCK, 0, .ELEM_TYPE);
893 0952
894 0953 +
895 0954 - If GETCONST found an error, increment our error counter
896 0955 otherwise convert the constant just read into a REAL
897 0956 quantity and store in CONSBLOCK[0].
898 0957
899 0958
```

```
900 0959 3 IF .B_CTYPE EQL K_NULL
901 0960 THEN
902 0961 B_ERR_FLAG = .B_ERR_FLAG + 1
903 0962 ELSE
904 0963
905 0964 IF NOT FOR$$CVT_TYPE (.B_CTYPE, L_CONSBLOCK, .ELEM_TYPE, CONSBLOCK [0])
906 0965 THEN
907 0966 B_ERR_FLAG = .B_ERR_FLAG + 1;
908 0967
909 0968 !+
910 0969 ! Must be a comma here.
911 0970 !-
912 0971
913 0972 IF NOT DELIM ( ) THEN B_ERR_FLAG = .B_ERR_FLAG + 1;
914 0973
915 0974 !+
916 0975 ! Get the imaginary part and convert it to a REAL quantity.
917 0976 ! Store the result into CONSBLOCK[1 or 2].
918 0977 !-
919 0978
920 0979 B_CTYPE = GETCONST (L_CONSBLOCK, 0, .ELEM_TYPE);
921 0980
922 0981 IF .B_CTYPE EQL K_NULL
923 0982 THEN
924 0983 B_ERR_FLAG = .B_ERR_FLAG + 1
925 0984 ELSE
926 0985
927 0986 IF .ELEM_TYPE NEQ DSC$K_DTYPE_H
928 0987 THEN
929 0988 BEGIN
930 0989
931 0990 IF NOT FOR$$CVT_TYPE (.B_CTYPE, L_CONSBLOCK, .ELEM_TYPE,
932 0991 IF .ELEM_TYPE EQL DSC$K_DTYPE_F THEN CONSBLOCK [1] ELSE CONSBLOCK [2])
933 0992 THEN
934 0993 B_ERR_FLAG = B_ERR_FLAG + 1;
935 0994
936 0995 END;
937 0996
938 0997 !+
939 0998 ! Skip blanks here. Better not be a comma!
940 0999 !-
941 1000
942 1001 IF DELIM ( ) THEN B_ERR_FLAG = .B_ERR_FLAG + 1;
943 1002
944 1003 !+
945 1004 ! Check for the required ')'
946 1005 !-
947 1006
948 1007
949 1008
950 1009 IF .(.CCB [LUB$A_BUF_PTR])<0, 8> EQL %C)'
951 1010 THEN
952 1011 CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1
953 1012 ELSE
954 1013 B_ERR_FLAG = .B_ERR_FLAG + 1;
955 1014
956 1015 !+
```

```

: 957      1016 3      ! If any errors occurred return NULL else COMPLEX type.
: 958      1017 3      !-
: 959      1018 3
: 960      1019 5      RETURN ((IF .B_ERR_FLAG EQL 0 THEN K_COMP ELSE
: 961      1020 6          BEGIN
: 962      1021 6          CCB [ISB$B_ERR_NO] = FOR$K_INPCONERR;
: 963      1022 6          K_NULL
: 964      1023 6          END
: 965      1024 3      ));
: 966      1025 2      END;
: 967      1026 2
: 968      1027 2
: 969      1028 2      !+ Logical constant.
: 970      1029 2      ! Point descriptor DSC to the field and set type to LOG.
: 971      1030 2      !-
: 972      1031 2
: 973      1032 2      [X'C'T', X'C'F', X'C't', X'C'f'] :
: 974      1033 2      BEGIN
: 975      1034 3      GETFIELD (DSC);
: 976      1035 3      CTYPE = K_LOG;
: 977      1036 2      END;
: 978      1037 2
: 979      1038 2
: 980      1039 2      !+ Possible logical constant. Check second character.
: 981      1040 2      !-
: 982      1041 2
: 983      1042 2      [X'C'.'] :
: 984      1043 3      BEGIN
: 985      1044 3      CTYPE = GETFIELD (DSC);
: 986      1045 3
: 987      1046 3      IF .DSC [DSC$W_LENGTH] GEQ 1
: 988      1047 3      THEN
: 989      1048 4          BEGIN
: 990      1049 4              LOCAL
: 991      1050 4                  C
: 992      1051 4                  ADR : REF BLOCK [1];      ! second character
: 993      1052 4                  ! address of second character
: 994      1053 4
: 995      1054 4                  ADR = 1 + .DSC [DSC$A_POINTER];
: 996      1055 4                  C = .ADR [0, 0, 8, 0];
: 997      1056 4
: 998      1057 4                  SELECT .C OF
: 999      1058 4                      SET
: 1000     1059 4
: 1001     1060 4                      [X'C'T', X'C't', X'C'F', X'C'f'] :
: 1002     1061 4                          CTYPE = K_LOG;
: 1003     1062 4                          TES;
: 1004     1063 4
: 1005     1064 4                      END
: 1006     1065 4
: 1007     1066 2                  END;
: 1008     1067 2
: 1009     1068 2
: 1010     1069 2      !+ Slash seen.
: 1011     1070 2      ! Set V_SLASH and return NULL value seen.
: 1012     1071 2      !-
: 1013     1072 2

```

```
1014 1073 2 [XC'/' ] :
1015 1074 2 BEGIN
1016 1075 2 CCB [ISBSV_SLASH] = 1;
1017 1076 2 RETURN K_NULL;
1018 1077 2 END;
1019 1078 2
1020 1079 2
1021 1080 2 + Comma or EOL.
1022 1081 2 Indicates null field. Return NULL value seen.
1023 1082 2 -
1024 1083 2
1025 1084 2 [XC' ' -1] :
1026 1085 2 RETURN K_NULL;
1027 1086 2
1028 1087 2
1029 1088 2 + String constant.
1030 1089 2 Gather up the string (handling double 's intelligently).
1031 1090 2 If STRINGFLAG is 1, store the string through CONSBLOCK[0].
1032 1091 2 The string returned is always 255 bytes long (blank padded).
1033 1092 2 If the string read is longer than 255 chars, SIGNAL [ISIO_SYN
1034 1093 2 and ignore the rest of the characters after the 255th.
1035 1094 2 -
1036 1095 2
1037 1096 2 [XC'''' ] :
1038 1097 2 BEGIN
1039 1098 2
1040 1099 2 LOCAL
1041 1100 2 C, ! Local character holder
1042 1101 2 A_BUF_PTR, ! if STRINGFLAG, points to callers buffer
1043 1102 2 A_BUF_END; ! if STRINGFLAG, points to end of buffer
1044 1103 2
1045 1104 2 +
1046 1105 2 Initialize locals
1047 1106 2 -
1048 1107 2
1049 1108 2 IF .STRINGFLAG
1050 1109 2 THEN
1051 1110 2 BEGIN
1052 1111 2 A_BUF_PTR = .CONSBLOCK [0];
1053 1112 2 A_BUF_END = .A_BUF_PTR + 255;
1054 1113 2 END;
1055 1114 2
1056 1115 2 C = NEXTCHAR;
1057 1116 2
1058 1117 2 +
1059 1118 2 Loop forever. Loop logic does an EXITLOOP when the
1060 1119 2 closing quote character is found.
1061 1120 2 -
1062 1121 2
1063 1122 2 WHILE 1 DO
1064 1123 2 BEGIN
1065 1124 2
1066 1125 2 +
1067 1126 2 If End-Of-Line is seen read another record, get the
1068 1127 2 first character and continue looping.
1069 1128 2 -
1070 1129 2
```

```

: 1071      1130  4      IF .C LSS 0
: 1072      1131  4      THEN
: 1073      1132  5      BEGIN
: 1074      1133  5      JSB_REC1 (FOR$$AA REC PR1 +
: 1075      1134  5      .FOR$$AA REC PR1 [CCB [ISB$B_STTM_TYPE] -
: 1076      1135  5      ISB$K FORSTTVLO + 1));
: 1077      1136  5      C = THISCRAR;
: 1078      1137  5      END
: 1079      1138  4      ELSE
: 1080      1139  5      BEGIN
: 1081      1140  5
: 1082      1141  5      IF .C EQL %C''''
: 1083      1142  5      THEN
: 1084      1143  6      BEGIN
: 1085      1144  6      C = NEXTCHAR;
: 1086      1145  6
: 1087      1146  6      IF .C NEQ %C'''' THEN EXITLOOP;
: 1088      1147  6
: 1089      1148  5      END;
: 1090      1149  5
: 1091      1150  5      !+
: 1092      1151  5      ! If the buffer just overflowed, SIGNAL LISIO_SYN
: 1093      1152  5      ! If the buffer overflowed sometime previous to this
: 1094      1153  5      ! iteration, don't do anything with the character.
: 1095      1154  5      ! Otherwise (less than 255 chars seen), continue to
: 1096      1155  5      ! Store bytes in the users buffer.
: 1097      1156  5      !-
: 1098      1157  5
: 1099      1158  5      IF .STRINGFLAG
: 1100      1159  5      THEN
: 1101      1160  6      BEGIN
: 1102      1161  6
: 1103      1162  7      IF (.A_BUF_PTR LSSA .A_BUF_END)
: 1104      1163  6      THEN
: 1105      1164  6      CH$WCHAR (.C, .A_BUF_PTR)
: 1106      1165  6      ELSE
: 1107      1166  6
: 1108      1167  6      IF (.A_BUF_PTR EQLA .A_BUF_END) THEN CCB [ISB$B_ERR_NO] = FOR$K_LISIO_SYN;
: 1109      1168  6
: 1110      1169  6      A_BUF_PTR = .A_BUF_PTR + 1;
: 1111      1170  5      END;
: 1112      1171  5
: 1113      1172  5      C = NEXTCHAR;
: 1114      1173  4      END;
: 1115      1174  4
: 1116      1175  3      END;                                ! End of main loop
: 1117      1176  3
: 1118      1177  3      !+
: 1119      1178  3      ! Blank pad the string to a length of 255 bytes.
: 1120      1179  3      !-
: 1121      1180  3
: 1122      1181  3      IF .STRINGFLAG
: 1123      1182  3      THEN
: 1124      1183  4      BEGIN
: 1125      1184  4      LOCAL
: 1126      1185  4      PAD_LENGTH;
: 1127      1186  4      PAD_LENGTH = .A_BUF_END - .A_BUF_PTR;
```

```

: 1128      1187      4          IF .PAD_LENGTH GTR 0 . Could be negative.
: 1129      1188      4          THEN
: 1130      1189      4              CH$FILL (%C' ', .PAD_LENGTH, .A_BUF_PTR);
: 1131      1190      3          END;
: 1132      1191      3
: 1133      1192      3          RETURN K_CHAR;
: 1134      1193      3          END;
: 1135      1194      3
: 1136      1195      3
: 1137      1196      3          !+
: 1138      1197      3          ! Right parenthesis. This can happen if a non-complex value was ended
: 1139      1198      3          ! with a right paren, since ')' is one of the possible value
: 1140      1199      3          ! separators. Give an error.
: 1141      1200      3          -
: 1142      1201      3          [')'] :
: 1143      1202      3              BEGIN
: 1144      1203      3              (CB [ISB$B ERR_NO] = FOR$K_LISIO_SYN;
: 1145      1204      3              RETURN K_NULL;
: 1146      1205      3              END;
: 1147      1206      3
: 1148      1207      3
: 1149      1208      3          !+
: 1150      1209      3          ! It's an integer or real constant (I hope).
: 1151      1210      3          ! Gather the constant and return its type.
: 1152      1211      3          -
: 1153      1212      3          [OTHERWISE] :
: 1154      1213      3              BEGIN
: 1155      1214      3              CTYPE = GETFIELD (DSC);
: 1156      1215      3
: 1157      1216      3              IF .DSC [DSC$W_LENGTH] EQL 0 THEN RETURN K_NULL;
: 1158      1217      3
: 1159      1218      3              END;
: 1160      1219      3          TES;
: 1161      1220      3
: 1162      1221      3          !+
: 1163      1222      3          ! Make special case for next (this) character being '*'. If so,
: 1164      1223      3          ! then this is a repeat count and must be an integer. If it isn't,
: 1165      1224      3          ! the convert will fail.
: 1166      1225      3          -
: 1167      1226      3
: 1168      1227      3          IF THISCHAR EQLU %C'*'
: 1169      1228      3          THEN
: 1170      1229      3              BEGIN
: 1171      1230      3
: 1172      1231      3              IF OTS$CVT_TI_L (DSC, CONSBLOCK [0]) THEN RETURN K_INT;
: 1173      1232      3
: 1174      1233      3          !+
: 1175      1234      3          ! If we get here, either the field wasn't an integer or it
: 1176      1235      3          ! got a conversion error. In either case, having a type of
: 1177      1236      3          ! K_NULL will cause an error eventually.
: 1178      1237      3          -
: 1179      1238      3
: 1180      1239      3              RETURN K_NULL;
: 1181      1240      3              END;
: 1182      1241      3
: 1183      1242      3
: 1184      1243      3          !+
: 1184      1243      2          ! Now that we have the LOG, INT, or REAL constant (as a string pointed
```

```

: 1185      1244  2      ! to by DSC), it must be converted into binary. Use the library
: 1186      1245  2      ! input conversion routines to store the resultant value into
: 1187      1246  2      ! CONSBLOCK[0]. Return the type of constant seen as routine value.
: 1188      1247  2      !
: 1189      1248  2
: 1190      1249  2      RETURN
: 1191      1250  3      BEGIN
: 1192      1251  3
: 1193      1252  3      IF NOT
: 1194      1253  4      BEGIN
: 1195      1254  4
: 1196      1255  4      SELECTONE .ELEM_TYPE OF
: 1197      1256  4      SET
: 1198      1257  4      [DSC$K_DTYPE_F] :
: 1199      1258  4      IF .CTYPE EQL K_LOG THEN OTSS$CVT_TL_L ELSE OTSS$CVT_T_F;
: 1200      1259  4
: 1201      1260  4      [DSC$K_DTYPE_D] :
: 1202      1261  4      IF .CTYPE EQL K_LOG THEN OTSS$CVT_TL_L ELSE OTSS$CVT_T_D;
: 1203      1262  4
: 1204      1263  4      [DSC$K_DTYPE_G] :
: 1205      1264  4      IF .CTYPE EQL K_LOG THEN OTSS$CVT_TL_L ELSE OTSS$CVT_T_G;
: 1206      1265  4
: 1207      1266  4      [DSC$K_DTYPE_H] :
: 1208      1267  4      IF .CTYPE EQL K_LOG THEN OTSS$CVT_TL_L ELSE OTSS$CVT_T_H;
: 1209      1268  4
: 1210      1269  4      [OTHERWISE] :
: 1211      1270  4      CASE .CTYPE FROM K_LOG TO K_REAL OF
: 1212      1271  4      SET
: 1213      1272  4      [K_LOG] :
: 1214      1273  4      OTSS$CVT_TL_L;
: 1215      1274  4
: 1216      1275  4      [K_INT] :
: 1217      1276  4      OTSS$CVT_TL_L;
: 1218      1277  4
: 1219      1278  4      [K_REAL] :
: 1220      1279  4      OTSS$CVT_T_D;
: 1221      1280  4
: 1222      1281  4      TES
: 1223      1282  4
: 1224      1283  4
: 1225      1284  4
: 1226      1285  4
: 1227      1286  4
: 1228      1287  4
: 1229      1288  4
: 1230      1289  4      TES
: 1231      1290  4
: 1232      1291  4      END
: 1233      1292  3      (DSC, CONSBLOCK [0])
: 1234      1293  3      THEN
: 1235      1294  4      BEGIN
: 1236      1295  4      CCB [ISB$B_ERR_NO] = FOR$K_INPCONERR;
: 1237      1296  4      K NULL
: 1238      1297  4      END
: 1239      1298  3      ELSE
: 1240      1299  3      .CTYPE
: 1241      1300  3
```


: 1242
: 1243

1301 2
1302 1

END;
END;

```

                                07FC 00000 GETCONST:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10
5A      0000V  CF  9E 00002  MOVAB DELIM, R10      : 0846
59      0000V  CF  9E 00007  MOVAB GETFIELD, R9
58 00000000G  00 9E 0000C  MOVAB OT$$CVT_I1_L, R8
57 00000000G  00 9E 00013  MOVAB FOR$$AA_REC_PR1, R7
5E      1C  C2 0001A  SUBL2 #28, SP
B4 AB      B0  AB  D1 0001D  CMPL  -80(CCB), -76(CCB) : 0917
      05  1F 00022  BLSSU 1$
      01  CE 00024  MNEGL #1, R2
      04  11 00027  BRB 2$
52      B0  BB  9A 00029 1$: MOVZBL @-80(CCB), R2
28      52  D1 0002D 2$: CMPL  R2, #40 : 0924
      03  13 00030  BEQL  3$
      00A1 31 00032  BRW 16$
      6E  94 00035 3$: CLRB  B_ERR_FLAG : 0936
      B0  AB  D6 00037  INCL  -80(CCB) : 0937
6A      00  FB 0003A  CALLS #0, DELIM : 0944
02      50  E9 0003D  BLBC  R0, 4$
      6E  96 00040  INCB  B_ERR_FLAG
52      OC  AC  D0 00042 4$: MOVL  E[EM_TYPE, R2 : 0951
      52  DD 00046  PUSHL R2
      7E  D4 00048  CLRL  -(SP)
AF AF      OC  AE  9F 0004A  PUSHAB L_CONSBLOCK
53      03  FB 0004D  CALLS #3, GETCONST
      50  90 00051  MOVB  R0, B_CTYPE
      13  13 00054  BEQL  5$ : 0959
      04  AC  DD 00056  PUSHL CONSBLOCK : 0964
      52  DD 00059  PUSHL R2
      OC  AE  9F 0005B  PUSHAB L_CONSBLOCK
FE45 7E      53  9A 0005E  MOVZBL B_CTYPE, -(SP)
CF      04  FB 00061  CALLS #4, FOR$$CVT_TYPE
02      50  E8 00066  BLBS  R0, 6$
      6E  96 00069 5$: INCB  B_ERR_FLAG : 0966
6A      00  FB 0006B 6$: CALLS #0, DELIM : 0972
02      50  E8 0006E  BLBS  R0, 7$
      6E  96 00071  INCB  B_ERR_FLAG
      52  DD 00073 7$: PUSHL R2 : 0979
      7E  D4 00075  CLRL  -(SP)
      OC  AE  9F 00077  PUSHAB L_CONSBLOCK
82 AF      03  FB 0007A  CALLS #3, GETCONST
53      50  90 0007E  MOVB  R0, B_CTYPE
      04  12 00081  BNEQ  8$ : 0981
      6E  96 00083  INCB  B_ERR_FLAG : 0983
      2F  11 00085  BRB 1$
      52  D1 00087 8$: CMPL  R2, #28 : 0986
      2A  13 0008A  BEQL  11$
      52  D1 0008C  CMPL  R2, #10 : 0992
      07  12 0008F  BNEQ  9$
50      04  AC  04  C1 00091  ADDL3 #4, CONSBLOCK, R0

```

50	04	AC	05	11	00096	BRB	10\$		
			08	C1	00098	9\$: ADDL3	#8, CONSBLOCK, R0		
			50	DD	0009D	10\$: PUSHL	R0		
			52	DD	0009F	PUSHL	R2		0990
			0C	AE	9F 000A1	PUSHAB	L_CONSBLOCK		
		7E	53	9A	000A4	MOVZBL	B_CTYPE, -(SP)		
	FDFE	CF	04	FB	000A7	CALLS	#4, FOR\$\$CVT_TYPE		
		07	50	E8	000AC	BLBS	R0, 11\$		
		50	01	AE	9E 000AF	MOVAB	B_ERR_FLAG+1, R0		0995
		6E	50	90	000B3	MOVB	R0, B_ERR_FLAG		
		6A	00	FB	000B6	11\$: CALLS	#0, DELIM		1003
		02	50	E9	000B9	BLBC	R0, 12\$		
			6E	96	000BC	INCB	B_ERR_FLAG		
		29	80	BB	91 000BE	12\$: CMPB	@-80(CCB), #41		1009
			05	12	000C2	BNEQ	13\$		
			80	AB	D6 000C4	INCL	-80(CCB)		1011
			02	11	000C7	BRB	14\$		
			6E	96	000C9	13\$: INCB	B_ERR_FLAG		1013
			6E	95	000CB	14\$: TSTB	B_ERR_FLAG		1019
			03	13	000CD	BEQL	15\$		
			01C2	31	000CF	BRW	52\$		
		50	04	D0	000D2	15\$: MOVL	#4, R0		
			04	000D5	RET				
00000046		8F	52	D1	000D6	16\$: CMPL	R2, #70		1032
			1B	13	000DD	BEQL	17\$		
00000054		8F	52	D1	000DF	CMPL	R2, #84		
			12	13	000E6	BEQL	17\$		
00000066		8F	52	D1	000E8	CMPL	R2, #102		
			09	13	000EF	BEQL	17\$		
00000074		8F	52	D1	000F1	CMPL	R2, #116		
			08	12	000F8	BNEQ	18\$		
			14	AE	9F 000FA	17\$: PUSHAB	DSC		1034
		69	01	FB	000FD	CALLS	#1, GETFIELD		
			3F	11	00100	BRB	19\$		1035
		2E	52	D1	00102	18\$: CMPL	R2, #46		1042
			40	12	00105	BNEQ	21\$		
			14	AE	9F 00107	PUSHAB	DSC		1044
		69	01	FB	0010A	CALLS	#1, GETFIELD		
		56	50	90	0010D	MOVB	R0, CTYPE		
			14	AE	B5 00110	TSTW	DSC		1046
			2F	13	00113	BEQL	20\$		
50		18	AE	01	C1 00115	ADDL3	#1, DSC+4, ADR		1054
			50	60	9A 0011A	MOVZBL	(ADR), C		1055
00000046		8F	50	D1	0011D	CMPL	C, #70		1060
			1B	13	00124	BEQL	19\$		
00000054		8F	50	D1	00126	CMPL	C, #84		
			12	13	0012D	BEQL	19\$		
00000066		8F	50	D1	0012F	CMPL	C, #102		
			09	13	00136	BEQL	19\$		
00000074		8F	50	D1	00138	CMPL	C, #116		
			03	12	0013F	BNEQ	20\$		
		56	01	90	00141	19\$: MOVB	#1, CTYPE		1061
			00B4	31	00144	20\$: BRW	39\$		1046
		2F	52	D1	00147	21\$: CMPL	R2, #47		1073
			07	12	0014A	BNEQ	23\$		
		96	10	88	0014C	BISB2	#16, -106(CCB)		1075
			014D	31	00150	22\$: BRW	54\$		1076

	FFFFFFF	8F		52	D1	00153	23\$:	CMPL	R2, #-1		1084
				F4	13	0015A		BEQL	22\$		
		2C		52	D1	0015C		CMPL	R2, #44		
				EF	13	0015F		BEQL	22\$		
		27		52	D1	00161		CMPL	R2, #39		1096
				7A	12	00164		BNEQ	36\$		
		53	08	AC	E9	00166		BLBC	STRINGFLAG, 32\$		1108
		52	04	BC	D0	0016A		MOVL	@CONSBLOCK, A_BUF_PTR		1111
		53	00FF	C2	9E	0016E		MOVAB	255(R2), A_BUF_END		1112
				48	11	00173		BRB	32\$		1115
		54	B0	BB	9A	00175	24\$:	MOVZBL	@-80(CCB), C		
				0E	18	00179	25\$:	BGEQ	26\$		1130
		50	FF71	CB	9A	0017B		MOVZBL	-143(CCB), R0		1135
		50		6740	D0	00180		MOVL	FOR\$\$AA_REC_PRI[R0], R0		1134
				6740	16	00184		JSB	FOR\$\$AA_REC_PRI[R0]		1133
				37	11	00187		BRB	33\$		1136
		27		54	D1	00189	26\$:	CMPL	C, #39		1141
				18	12	0018C		BNEQ	29\$		
			B0	AB	D6	0018E		INCL	-80(CCB)		1144
	B4	AB	B0	AB	D1	00191		CMPL	-80(CCB), -76(CCB)		
				05	1F	00196		BLSSU	27\$		
		54		01	CE	00198		MNEGL	#1, C		
				04	11	0019B		BRB	28\$		
		54	B0	BB	9A	0019D	27\$:	MOVZBL	@-80(CCB), C		
		27		54	D1	001A1	28\$:	CMPL	C, #39		1146
				26	12	001A4		BNEQ	34\$		
		13	08	AC	E9	001A6	29\$:	BLBC	STRINGFLAG, 32\$		1158
		53		52	D1	001AA		CMPL	A_BUF_PTR, A_BUF_END		1162
				05	1E	001AD		BGEQU	30\$		
		62		54	90	001AF		MOVB	C, (A_BUF_PTR)		1164
				07	11	001B2		BRB	31\$		
				05	12	001B4	30\$:	BNEQ	31\$		1167
	FF70	CB		3B	90	001B6		MOVB	#59, -144(CCB)		
				52	D6	001BB	31\$:	INCL	A_BUF_PTR		1169
			B0	AB	D6	001BD	32\$:	INCL	-80(CCB)		1172
	B4	AB	B0	AB	D1	001C0	33\$:	CMPL	-80(CCB), -76(CCB)		
				AE	1F	001C5		BLSSU	24\$		
		54		01	CE	001C7		MNEGL	#1, C		
				AD	11	001CA		BRB	25\$		
		0C	08	AC	E9	001CC	34\$:	BLBC	STRINGFLAG, 35\$		1181
	50	53		52	C3	001D0		SUBL3	A_BUF_PTR, A_BUF_END, PAD_LENGTH		1186
				06	15	001D4		BLEQ	35\$		1187
50		20		00	2C	001D6		MOVCS	#0, (SP), #32, PAD_LENGTH, (A_BUF_PTR)		1189
				62		001DB					
		50		05	D0	001DC	35\$:	MOVL	#5, R0		1192
				04		001DF		RET			
		29		52	D1	001E0	36\$:	CMPL	R2, #41		1201
				08	12	001E3		BNEQ	38\$		
				3B	90	001E5		MOVB	#59, -144(CCB)		1203
	FF70	CB		00B3	31	001EA	37\$:	BRW	54\$		1204
			14	AE	9F	001ED	38\$:	PUSHAB	DSC		1214
		69		01	FB	001F0		CALLS	#1, GETFIELD		
		56		50	90	001F3		MOVB	R0, CTYPE		
			14	AE	B5	001F6		TSTW	DSC		1216
				EF	13	001F9		BEQL	37\$		
	B4	AB	B0	AB	D1	001FB	39\$:	CMPL	-80(CCB), -76(CCB)		1227
				05	1F	00200		BLSSU	40\$		

50		01	CE	00202		MNEGL	#1, R0		
		04	11	00205		BRB	41\$		
50	B0	BB	9A	00207	40\$:	MOVZBL	@-80(CCB), R0		
2A		50	D1	0020B	41\$:	CMPL	R0, #42		
		10	12	0020E		BNEQ	42\$		
	04	AC	DD	00210		PUSHL	CONSBLOCK	1231	
	18	AE	9F	00213		PUSHAB	DSC		
68		02	FB	00216		CALLS	#2, OTSS\$CVT_TI_L		
CE		50	E9	00219		BLBC	R0, 37\$		
50		02	D0	0021C		MOVL	#2, R0		
		04	00	0021F		RET			
50	0C	AC	D0	00220	42\$:	MOVL	ELEM_TYPE, R0	1255	
0A		50	D1	00224		CMPL	R0, #10	1258	
		0E	12	00227		BNEQ	43\$		
01		56	91	00229		CMPB	CTYPE, #1	1260	
		45	13	0022C		BEQL	48\$		
50	00000000G	00	9E	0022E		MOVAB	OTSS\$CVT_T_F, R0		
		51	11	00235		BRB	51\$		
0B		50	D1	00237	43\$:	CMPL	R0, #11	1262	
		07	12	0023A		BNEQ	44\$		
01		56	91	0023C		CMPB	CTYPE, #1	1264	
		32	13	0023F		BEQL	48\$		
		3E	11	00241		BRB	50\$		
1B		50	D1	00243	44\$:	CMPL	R0, #27	1266	
		0E	12	00246		BNEQ	45\$		
01		56	91	00248		CMPB	CTYPE, #1	1268	
		26	13	0024B		BEQL	48\$		
50	00000000G	00	9E	0024D		MOVAB	OTSS\$CVT_T_G, R0		
		32	11	00254		BRB	51\$		
1C		50	D1	00256	45\$:	CMPL	R0, #28	1270	
		0E	12	00259		BNEQ	46\$		
01		56	91	0025B		CMPB	CTYPE, #1	1272	
		13	13	0025E		BEQL	48\$		
50	00000000G	00	9E	00260		MOVAB	OTSS\$CVT_T_H, R0		
		1F	11	00267		BRB	51\$		
	02	01	56	8F	46\$:	CASEB	CTYPE, #1, #2	1276	
	0014	000F	0006	0026D	47\$:	.WORD	48\$-47\$,- 49\$-47\$,- 50\$-47\$,-		
50	00000000G	00	9E	00273	48\$:	MOVAB	OTSS\$CVT_TL_L, R0		
		0C	11	0027A		BRB	51\$		
50		68	9E	0027C	49\$:	MOVAB	OTSS\$CVT_TI_L, R0		
		07	11	0027F		BRB	51\$		
50	00000000G	00	9E	00281	50\$:	MOVAB	OTSS\$CVT_T_D, R0		
		04	AC	DD	51\$:	PUSHL	CONSBLOCK	1292	
		18	AE	9F		PUSHAB	DSC		
60		02	FB	0028E		CALLS	#2, (R0)		
08		50	E8	00291		BLBS	R0, 53\$		
FF70	CB	40	8F	90	52\$:	MOVB	#64, -144(CCB)	1295	
		04	11	0029A		BRB	54\$	1294	
50		56	9A	0029C	53\$:	MOVZBL	CTYPE, R0	1299	
		04	00	0029F		RET		1250	
		50	D4	002A0	54\$:	CLRL	R0	1302	
		04	00	002A2		RET			

; Routine Size: 675 bytes, Routine Base: _FOR\$CODE + 0387

FOR\$\$UDF_RL
1-025

FORTRAN list-directed input, UDF level
GETCONST

L 8
16-Sep-1984 00:47:40
14-Sep-1984 12:32:51

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFRL.B32;1

Page 35
(7)

FOR
1-C

```
1245 1303 1 %SBTTL 'GETFIELD'
1246 1304 1 ROUTINE GETFIELD (DSC) : CALL_CCB =
1247 1305 1
1248 1306 1 |++
1249 1307 1 | FUNCTIONAL DESCRIPTION:
1250 1308 1 |
1251 1309 1 |     Determine the length and type of the field pointed to by LUB$A_BUF_PTR.
1252 1310 1 |     Point string descriptor DSC to the field.
1253 1311 1 |     Return the type as the routine value.
1254 1312 1 |
1255 1313 1 | FORMAL PARAMETERS:
1256 1314 1 |
1257 1315 1 |     DSC.wl.r           String descriptor to point to field
1258 1316 1 |
1259 1317 1 | IMPLICIT INPUTS:
1260 1318 1 |
1261 1319 1 |     NONE
1262 1320 1 |
1263 1321 1 | IMPLICIT OUTPUTS:
1264 1322 1 |
1265 1323 1 |     NONE
1266 1324 1 |
1267 1325 1 | ROUTINE VALUE:
1268 1326 1 |
1269 1327 1 |     Returns the type of constant seen (as a small integer).
1270 1328 1 |
1271 1329 1 | COMPLETION CODES:
1272 1330 1 |
1273 1331 1 |     NONE
1274 1332 1 |
1275 1333 1 | SIDE EFFECTS:
1276 1334 1 |
1277 1335 1 |     NONE
1278 1336 1 |
1279 1337 1 | --
1280 1338 1 |
1281 1339 2 | BEGIN
1282 1340 2 |
1283 1341 2 | EXTERNAL REGISTER
1284 1342 2 |     CCB : REF $FOR$CCB_DECL;
1285 1343 2 |
1286 1344 2 | MAP
1287 1345 2 |     DSC : REF BLOCK [8, BYTE];
1288 1346 2 |
1289 1347 2 | LOCAL
1290 1348 2 |     T,           ! type of constant seen
1291 1349 2 |     C;           ! local character holder
1292 1350 2 |
1293 1351 2 | |++
1294 1352 2 | | Point the descriptor pointer to the start of the field.
1295 1353 2 | |--
1296 1354 2 |
1297 1355 2 | DSC [DSC$A_POINTER] = .(CCB [LUB$A_BUF_PTR]);
1298 1356 2 |
1299 1357 2 | |++
1300 1358 2 | | Assume type REAL
1301 1359 2 | |--
```

```

: 1302      1360  2
: 1303      1361  2
: 1304      1362  2      T = K PEAL;
: 1305      1363  2      C = TRISCHAR;
: 1306      1364  2
: 1307      1365  2      !+
: 1308      1366  2      ! Skip through the string looking for delimiters.
: 1309      1367  2      ! If a delimiter is seen,
: 1310      1368  2      ! or we hit EOL, we've reached the end of the constant.
: 1311      1369  2      !-
: 1312      1370  2      WHILE .C GEQ 0 DO
: 1313      1371  3          BEGIN
: 1314      1372  3
: 1315      1373  3          IF NOT CH$FAIL (CH$FIND_CH (6, UPLIT ('          ,/*)'), .C)) THEN EXITLOOP;
: 1316      1374  3
: 1317      1375  3          C = NEXTCHAR;
: 1318      1376  2          END;
: 1319      1377  2
: 1320      1378  2      DSC [DSC$W_LENGTH] = CH$DIFF (.CCB [LUB$A_BUF_PTR], .DSC [DSC$A_POINTER]);
: 1321      1379  2      RETURN .T;
: 1322      1380  1      END;

```

```

00 00 29 2A 2F 2C 09 20 0062A .BLKB 2
0062C P.AAB: .ASCII \ \<9>\,/*)\<0><0> ;

```

```

                                001C 00000 GETFIELD:
                                .WORD      Save R2,R3,R4
                                MOVL      DSC, R2
                                MOVL      -80(CCB), 4(R2)
                                MOVL      #3, T
                                BRB       4$
                                BRB       @-80(CCB), C
                                DD      AF  06      1C 19 00014 2$: BLSS      5$
                                DD      AF  06      53 3A 00016      LOCC      C, #6, P.AAB
                                DD      AF  06      02 12 0001B      BNEQ      3$
                                DD      AF  06      51 D4 0001D      CLRL      R1
                                DD      AF  06      51 D5 0001F 3$: TSTL      R1
                                DD      AF  06      0F 12 00021      BNEQ      5$
                                DD      AF  06      B4 AB      B0 AB D6 00023      INCL      -80(CCB)
                                DD      AF  06      B4 AB      B0 AB D1 00026 4$: CMPL      -80(CCB), -76(CCB)
                                DD      AF  06      53      E3 1F 0002B      BLSSU     1$
                                DD      AF  06      53      01 CE 0002D      MNEGL     #1, C
                                DD      AF  06      62 B0 AB      E2 11 00030      BRB       2$
                                DD      AF  06      50      A2 A3 00032 5$: SUBW3     4(R2), -80(CCB), (R2)
                                DD      AF  06      50      54 D0 00038      MOVL      T, R0
                                DD      AF  06      50      04 0003B      RET

```

; Routine Size: 60 bytes, Routine Base: _FOR\$CODE + 0634

```

1324 1381 1 %SBTTL 'LCL_HANDLER'
1325 1382 1 ROUTINE LCL_HANDLER (
1326 1383 1     SIG_ARGS
1327 1384 1     MECH_ARGS
1328 1385 1 ) =
1329 1386 1
1330 1387 1 ++
1331 1388 1 FUNCTIONAL DESCRIPTION:
1332 1389 1
1333 1390 1     Resignal Access Violation, otherwise call LIB$$SIG_TO_RET.
1334 1391 1
1335 1392 1 FORMAL PARAMETERS:
1336 1393 1
1337 1394 1     SIG_ARGS
1338 1395 1     MECH_ARGS
1339 1396 1
1340 1397 1 IMPLICIT INPUTS:
1341 1398 1
1342 1399 1     NONE
1343 1400 1
1344 1401 1 IMPLICIT OUTPUTS:
1345 1402 1
1346 1403 1     NONE
1347 1404 1
1348 1405 1 COMPLETION CODES:
1349 1406 1
1350 1407 1     Will return any error other than Access Violation as a status
1351 1408 1
1352 1409 1 SIDE EFFECTS:
1353 1410 1
1354 1411 1     Resignals Access Violation
1355 1412 1
1356 1413 1 --
1357 1414 1
1358 1415 2 BEGIN
1359 1416 2 MAP
1360 1417 2     SIG_ARGS : REF BLOCK [, BYTE];
1361 1418 2 +
1362 1419 2 Check to see if the error is Access Violation. If it is, resignal so that it
1363 1420 2 is reported with the proper PC and PSL. Otherwise, return all other errors as
1364 1421 2 statuses.
1365 1422 2 -
1366 1423 2     IF .SIG_ARGS [CHF$SIG_NAME] NEQ S$$_ACCVIO THEN LIB$$SIG_TO_RET(SIG_ARGS, MECH_ARGS);
1367 1424 2 +
1368 1425 2 LIB$$SIG_TO_RET will not return to this routine. It changes the error signal
1369 1426 2 to a return status and unwinds to the caller of the establisher of this handler.
1370 1427 2 -
1371 1428 2 RETURN S$$_RESIGNAL
1372 1429 1 END; ! Routine LCL_HANDLER

```

```

0000 0000 LCL_HANDLER:
50      04 AC D0 0002      .WORD Save nothing
                          MOVL SIG_ARGS, R0      : 1382
                                          : 1423

```


FOR\$\$UDF_RL
1-025

FORTLAN list-directed input, UDF level
LCL_HANDLER

C 9
16-Sep-1984 00:47:40
14-Sep-1984 12:32:51

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFRL.B32;1

Page 39
(9)

	0C		04	A0	D1	00006			CMP	4(R0), #12			
				0D	13	0000A			BEQ	1\$			
				08	AC	9F	0000C		PUSH	MECH_ARGS			
				04	AC	9F	0000F		PUSH	SIG_ARGS			
00000000G		00		02	FB	00012			CALL	#2, LIB\$SIG_TO_RET			
		50	0918	8F	3C	00019	1\$:		MOVZ	#2328, R0		1428	
				04	00	0001E			RET			1429	

; Routine Size: 31 bytes, Routine Base: _FOR\$CODE + 0670

```
1374 1430 1 %SBTTL'SKIPBLANKS'  
1375 1431 1 ROUTINE SKIPBLANKS : CALL_CCB =  
1376 1432 1  
1377 1433 1 ++  
1378 1434 1 FUNCTIONAL DESCRIPTION:  
1379 1435 1  
1380 1436 1     Skip over blanks, tabs, and EOLs and return the first  
1381 1437 1     'real' character.  
1382 1438 1  
1383 1439 1 FORMAL PARAMETERS:  
1384 1440 1  
1385 1441 1     NONE  
1386 1442 1  
1387 1443 1 IMPLICIT INPUTS:  
1388 1444 1  
1389 1445 1     LUB$A_BUF_PTR           points to first char to scan  
1390 1446 1  
1391 1447 1 IMPLICIT OUTPUTS:  
1392 1448 1  
1393 1449 1     NONE  
1394 1450 1  
1395 1451 1 ROUTINE VALUE:  
1396 1452 1  
1397 1453 1     Return the first real char found.  
1398 1454 1  
1399 1455 1 COMPLETION CODES:  
1400 1456 1  
1401 1457 1     NONE  
1402 1458 1  
1403 1459 1 SIDE EFFECTS:  
1404 1460 1  
1405 1461 1     Will cause a record to be read if no data is found in the current  
1406 1462 1     record.  
1407 1463 1  
1408 1464 1 --  
1409 1465 1  
1410 1466 2 BEGIN  
1411 1467 2  
1412 1468 2 EXTERNAL REGISTER  
1413 1469 2     CCB : REF $FOR$CCB_DECL;  
1414 1470 2  
1415 1471 2 LOCAL  
1416 1472 2     C;  
1417 1473 2  
1418 1474 2 WHILE 1 DO  
1419 1475 3 BEGIN  
1420 1476 3     C = THISCHAR;  
1421 1477 3  
1422 1478 3     WHILE .C EQL %C' ' OR .C EQL K_TAB DO  
1423 1479 3         C = NEXTCHAR;  
1424 1480 3  
1425 1481 3     IF .C GEQ 0 THEN RETURN .C;  
1426 1482 3  
1427 1483 3     JSB_REC1 (FOR$$AA_REC_PRI + .FOR$$AA_REC_PRI [.CCB [ISB$B_STM_TYPE] -  
1428 1484 3         ISB$K_FORSTTY[0 + 1]);  
1429 1485 2 END;  
1430 1486 2
```

: 1431
: 1432
1487 2 RETURN (0);
1488 1 END;

		003C 00000		SKIPBLANKS:		
53	00000000G	00	9E 00002	.WORD	Save R2,R3,R4,R5	: 1431
		11	11 00009	MOVAB	FOR\$\$AA_REC_PR1, R3	: 1476
52	B0	BB	9A 0000B	BRB	4\$: 1478
20		52	D1 0000F	MOVZBL	@-80(CCB), C	
		05	13 00012	CMPL	C, #32	
09		52	D1 00014	BEQL	3\$	
		0F	12 00017	CMPL	C, #9	
		B0	AB D6 00019	BNEQ	5\$	
B4	AB	B0	AB D1 0001C	INCL	-80(CCB)	: 1479
		E8	1F 00021	CMPL	-80(CCB), -76(CCB)	
52		01	CE 00023	BLSSU	1\$	
		E7	11 00026	MNEGL	#1, C	
		52	D5 00028	BRB	2\$	
		04	19 0002A	TSTL	C	: 1481
50		52	D0 0002C	BLSS	6\$	
			04 0002F	MOVL	C, R0	
50	FF71	CB	9A 00030	RET		
50		6340	D0 00035	MOVZBL	-143(CCB), R0	: 1484
		6340	16 00039	MOVL	FOR\$\$AA_REC_PR1[R0], R0	: 1483
		DE	11 0003C	JSB	FOR\$\$AA_REC_PR1[R0]	: 1474
				BRB	4\$	

: Routine Size: 62 bytes. Routine Base: _FOR\$CODE + 068F

```

: 1434 1489 1 %SBTTL'DELIM'
: 1435 1490 1 ROUTINE DELIM : CALL_CCB =
: 1436 1491 1
: 1437 1492 1 !++
: 1438 1493 1 FUNCTIONAL DESCRIPTION:
: 1439 1494 1
: 1440 1495 1 Process blanks, tabs, EOLs and commas.
: 1441 1496 1
: 1442 1497 1 FORMAL PARAMETERS:
: 1443 1498 1
: 1444 1499 1 NONE
: 1445 1500 1
: 1446 1501 1 IMPLICIT INPUTS:
: 1447 1502 1
: 1448 1503 1 LUB$A_BUF_PTR points to first char to scan
: 1449 1504 1
: 1450 1505 1 IMPLICIT OUTPUTS:
: 1451 1506 1
: 1452 1507 1 NONE
: 1453 1508 1
: 1454 1509 1 ROUTINE VALUE:
: 1455 1510 1
: 1456 1511 1 1 if 1 comma encountered
: 1457 1512 1 0 if no commas or 2 commas (null field)
: 1458 1513 1 COMPLETION CODES:
: 1459 1514 1
: 1460 1515 1 NONE
: 1461 1516 1
: 1462 1517 1 SIDE EFFECTS:
: 1463 1518 1
: 1464 1519 1 NONE
: 1465 1520 1
: 1466 1521 1 --
: 1467 1522 1
: 1468 1523 2 BEGIN
: 1469 1524 2
: 1470 1525 2 EXTERNAL REGISTER
: 1471 1526 2 CCB : REF $FOR$CCB_DECL;
: 1472 1527 2
: 1473 1528 2 IF SKIPBLANKS () NEQ %C', ' THEN RETURN 0;
: 1474 1529 2
: 1475 1530 2 (CB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1;
: 1476 1531 2 RETURN (SKIPBLANKS () NEQ %C', '));
: 1477 1532 1 END;

```

			0000 0000	DELIM:	.WORD	Save nothing	: 1490
BC	AF		00 FB 00002		CALLS	#0, SKIPBLANKS	: 1528
	2C		50 D1 00006		CMPL	R0, #44	:
			14 12 00009		BNEQ	2\$:
		B0	AB D6 0000B		INCL	-80(CCB)	: 1530
B0	AF		00 FB 0000E		CALLS	#0, SKIPBLANKS	: 1531
			51 D4 00012		CLRL	R1	:
	2C		50 D1 00014		CMPL	R0, #44	:

FOR\$\$UDF_RL
1-025

FORTTRAN List-directed input, UDF level
DELIM

G 9
16-Sep-1984 00:47:40
14-Sep-1984 12:32:51

VAX-11 Bliss-32 V4.0-742
[FORRTL.SRC]FORUDFRL.B32;1

Page 43
(11)

FOR
1-0

	02	13	00017		BEQL	1\$	
	51	D6	00019		INCL	R1	
50	51	D0	0001B	1\$:	MOVL	R1, R0	
		04	0001E		RET		
50	D4	0001F	2\$:	CLRL	R0		
	04	00021		RET			

1532

: Routine Size: 34 bytes, Routine Base: _FOR\$CODE + 06CD

: 1478 1533 1 END
: 1479 1534 1
: 1480 1535 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$CODE	1775	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	14	0	581	00:01.0
_\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	185	26	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\$:FORUDFRL/OBJ=OBJ\$:FORUDFRL MSRC\$:FORUDFRL/UPDATE=(ENHS\$:FORUDFRL)

: Size: 1644 code + 131 data bytes
: Run Time: 00:35.5
: Elapsed Time: 01:16.8
: Lines/CPU Min: 2592
: Lexemes/CPU-Min: 16652
: Memory Used: 262 pages
: Compilation Complete

