


```
FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF RRRRRRRR NN NN
FFFFFFFFF 000000 RRRRRRRR UU UU DDDDDDDD FFFFFFFFFF RRRRRRRR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF RRRRRRRR NN NN
FFFFFFFF 00 00 RRRRRRRR UU UU DD DD FFFFFFFF RRRRRRRR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 00 00 RR RR RR UU UU DD DD FF RR RR NN NN
FF 000000 RR RR UUUUUUUUU DDDDDDDD FFF RR RR NN NN
FF 000000 RR RR UUUUUUUUU DDDDDDDD FF RR RR NN NN
```

```
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_RN (%TITLE 'NAMELIST input UDF level'  
2 0002 0 IDENT = '1-005' ! File: FORUDFRN.B32 Edit: SBL1005  
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 Language Support  
32 0032 1  
33 0033 1 ABSTRACT:  
34 0034 1  
35 0035 1 This module contains the User Data Formatter level code for  
36 0036 1 FORTRAN NAMELIST input.  
37 0037 1  
38 0038 1 ENVIRONMENT: Runs at any access mode - AST reentrant  
39 0039 1  
40 0040 1 AUTHOR: Steven B. Lionel, CREATION DATE: 11-July-1980  
41 0041 1  
42 0042 1 MODIFIED BY:  
43 0043 1  
44 0044 1 1-001 - Original. SBL 11-July-1980  
45 0045 1 1-002 - Correct call for syntax error. Remove declaration of CALLG, not used. SBL 18-Nov-1980  
46 0046 1 1-003 - Add diagram of NAMELIST descriptor block. SBL 15-April-1981  
47 0047 1 *** First post-V3.0 edit ***  
48 0048 1 1-004 - Add query feature. Use prologue file. SBL 10-Jan-1983  
49 0049 1 1-005 - Allow group name to end with <TAB>. SPR 11-57564 SBL 2-Jun-1983  
50 0050 1 --  
51 0051 1
```

```
53 0052 1 %SBTTL 'Declarations'
54 0053 1
55 0054 1 PROLOGUE FILE:
56 0055 1
57 0056 1
58 0057 1 REQUIRE 'RTLIN:FORPROLOG';          ! FOR$ definitions
59 0123 1
60 0124 1
61 0125 1 TABLE OF CONTENTS:
62 0126 1
63 0127 1
64 0128 1 FORWARD ROUTINE
65 0129 1     FOR$$UDF_RN0 : JSB_UDF0 NOVALUE,      ! Start NAMELIST processing
66 0130 1     PROCESS_LIST : CALL_CCB NOVALUE,      ! Do bulk of NAMELIST processing
67 0131 1     FOR$$UDF_RN9 : JSB_UDF9 NOVALUE;      ! End NAMELIST processing
68 0132 1
69 0133 1
70 0134 1 INCLUDE FILES:
71 0135 1
72 0136 1
73 0137 1 LIBRARY 'RTLTPAMAC';          ! TPARSE macro definitions
74 0138 1
75 0139 1
76 0140 1 MACROS:
77 0141 1
78 0142 1     NONE
79 0143 1
80 0144 1 EQUATED SYMBOLS:
81 0145 1
82 0146 1     NONE
83 0147 1
84 0148 1 OWN STORAGE:
85 0149 1
86 0150 1     NONE
87 0151 1
88 0152 1 EXTERNAL REFERENCES:
89 0153 1
90 0154 1
91 0155 1 EXTERNAL ROUTINE
92 0156 1     FOR$$DO_NML_OUTPUT: CALL_CCB NOVALUE,      ! Do NAMELIST output
93 0157 1     FOR$$REC_RSNO : JSB_RECO NOVALUE,          ! Read a record
94 0158 1     FOR$$REC_WSNO : JSB_RECO NOVALUE,          ! Start a write
95 0159 1     FOR$$SIGNAL_STO: NOVALUE,
96 0160 1     LIB$TPARSE;          ! Parse input records
97 0161 1
98 0162 1 EXTERNAL
99 0163 1     FOR$$A_NMLSTATE,          ! TPARSE state tables
100 0164 1     FOR$$A_NMLKEYWD;      ! TPARSE keyword block
101 0165 1
```


FOR\$\$UDF_RN
1-005

NAMLIST input UDF level
FOR\$\$UDF_RNO - NAMLIST input UDF level

K 9
16-Sep-1984 00:49:15
14-Sep-1984 12:32:51

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

Page 3
(3)

```

103 0166 1 %SBTTL 'FOR$$UDF_RNO - NAMLIST input UDF level'
104 0167 1 GLOBAL ROUTINE FOR$$UDF_RNO: JSB_UDF0 NOVALUE
105 0168 1
106 0169 1
107 0170 1 ++
108 0171 1 FUNCTIONAL DESCRIPTION:
109 0172 1
110 0173 1 Initialize NAMLIST input.
111 0174 1
112 0175 1 CALLING SEQUENCE:
113 0176 1
114 0177 1 JSB FOR$$UDF_RNO
115 0178 1
116 0179 1 FORMAL PARAMETERS:
117 0180 1
118 0181 1 NONE
119 0182 1
120 0183 1 IMPLICIT INPUTS:
121 0184 1
122 0185 1 CCB Pointer to current logical unit block
123 0186 1
124 0187 1 IMPLICIT OUTPUTS:
125 0188 1
126 0189 1 NONE
127 0190 1
128 0191 1 COMPLETION STATUS:
129 0192 1
130 0193 1 NONE
131 0194 1
132 0195 1 SIDE EFFECTS:
133 0196 1
134 0197 1 See PROCESS_LIST.
135 0198 1
136 0199 1 --
137 0200 1
138 0201 2 BEGIN
139 0202 2
140 0203 2 EXTERNAL REGISTER
141 0204 2 CCB : REF $FOR$CCB_DECL;
142 0205 2
143 0206 2 !+
144 0207 2 ! Call PROCESS_LIST to do all the NAMLIST processing. Note that
145 0208 2 ! this is done from the initialization call since there are no element
146 0209 2 ! transmit calls for NAMLIST.
147 0210 2 !-
148 0211 2
149 0212 2 PROCESS_LIST ();
150 0213 2
151 0214 2 RETURN;
152 0215 1 END;

! End of routine FOR$$UDF_RNO
```

```

.TITLE FOR$$UDF_RN NAMLIST input UDF level
.IDENT \1-005\

.EXTRN FOR$$DO_NML_OUTPUT
.EXTRN FOR$$REC_RSNO, FOR$$REC_WSNO
```

FOR\$\$UDF_RN
1-005

NAMLIST input UDF level
FOR\$\$UDF_RN0 - NAMLIST input UDF level

L 9
16-Sep-1984 00:49:15
14-Sep-1984 12:32:51

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

Page 4
(3)

.EXTRN FOR\$\$SIGNAL_STO
.EXTRN LIB\$TPARSE, -FOR\$\$A_NMLSTATE
.EXTRN FOR\$\$A_NMLKEYWD

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

0000V CF

00 FB 00000 FOR\$\$UDF_RN0::

05 00005

CALLS #0, PROCESS_LIST
RSB

: 0212
: 0215

; Routine Size: 6 bytes, Routine Base: _FOR\$CODE + 0000

; 153 0216 1 !<BLF/PAGE>


```
155 0217 1 %SBTTL 'PROCESS_LIST - Process NAMELIST input'
156 0218 1 ROUTINE PROCESS_LIST ! Process NAMELIST input
157 0219 1 : CALL_CCB NOVALUE =
158 0220 1
159 0221 1 ++
160 0222 1 FUNCTIONAL DESCRIPTION:
161 0223 1
162 0224 1 Process NAMELIST input. Read records from the current unit until a
163 0225 1 NAMELIST group with the specified name is found. Process NAMELIST
164 0226 1 assignments to user variables until the group end is found.
165 0227 1
166 0228 1 CALLING SEQUENCE:
167 0229 1
168 0230 1 CALL PROCESS_LIST
169 0231 1
170 0232 1 FORMAL PARAMETERS:
171 0233 1
172 0234 1 NONE
173 0235 1
174 0236 1 IMPLICIT INPUTS:
175 0237 1
176 0238 1 CCB Pointer to current logical unit block
177 0239 1 ISB$A_FMT_BEG Pointer to NAMELIST descriptor block
178 0240 1
179 0241 1 IMPLICIT OUTPUTS:
180 0242 1
181 0243 1 NONE
182 0244 1
183 0245 1 COMPLETION STATUS:
184 0246 1
185 0247 1 NONE
186 0248 1
187 0249 1 SIDE EFFECTS:
188 0250 1
189 0251 1 Reads one or more records from the input file. Performs zero or more
190 0252 1 user variable assignments.
191 0253 1
192 0254 1 --
193 0255 1
194 0256 1 !<BLF/PAGE>
```

```
196 0257 1 ++
197 0258 1 Each NAMELIST descriptor block has the following form:
198 0259 1
199 0260 1      3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
200 0261 1      1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
201 0262 1
202 0263 1
203 0264 1 0 +-----+
204 0265 1 | Address of ASCIC name of NAMELIST group |
205 0266 1 +-----+
206 0267 1 1 | Reserved | Count of NAMELIST variables |
207 0268 1 +-----+
208 0269 1 2 | Address of ASCIC name of variable 1 |
209 0270 1 +-----+
210 0271 1 3 | Address of standard VAX descriptor for variable 1 |
211 0272 1 +-----+
212 0273 1 4 | |
213 0274 1 | |
214 0275 1 5 | Address of ASCIC name of variable n |
215 0276 1 +-----+
216 0277 1 6 | Address of standard VAX descriptor for variable n |
217 0278 1 +-----+
218 0279 1
219 0280 1 The NAMELIST group name and the variable names which are pointed to in
220 0281 1 the NAMELIST descriptor block are upper case only. The FORTRAN
221 0282 1 compiler or other calling program is responsible for case conversion
222 0283 1 of the name strings. In NAMELIST input data, case is significant only
223 0284 1 in character literals. The run-time library is responsible for case
224 0285 1 conversion of NAMELIST input data.
225 0286 1
226 0287 1 The allowable data types in variable descriptors are BU (BYTE), WU,
227 0288 1 LU, W, L, F, D, G, H, T, FC, DC, and GC. The allowable descriptor
228 0289 1 classes are scalar and array. For the array class descriptor, the
229 0290 1 descriptor flags COLUMN, COEFF, and BOUNDS must be set, indicating
230 0291 1 column-major order and the presence of coefficient and bounds blocks.
231 0292 1 The number of dimensions must not exceed 7.
232 0293 1 --
233 0294 1
234 0295 1 !<BLF/PAGE>
```



```
236 0296 2 BEGIN
237 0297 2
238 0298 2 EXTERNAL REGISTER
239 0299 2 CCB : REF $FOR$CCB_DECL;
240 0300 2
241 0301 2 LOCAL
242 0302 2 PARAM_BLOCK : BLOCK [NML$K_BLKLENGTH, BYTE] FIELD (NML$FIELDS);
243 0303 2
244 0304 2 !+
245 0305 2 ! Fill in basic fields of TPARSE parameter block
246 0306 2 !-
247 0307 2
248 0308 2 CH$FILL (0, NML$K_BLKLENGTH, PARAM_BLOCK); ! Fill with zeroes
249 0309 2 PARAM_BLOCK [TPAS$_COUNT] = TPAS$_COUNT;
250 0310 2 PARAM_BLOCK [NML$_LISTBLOCK] = .CCB [ISB$_FMT_BEG]; ! NAMELIST block address
251 0311 2 PARAM_BLOCK [NML$_CCB] = .CCB; ! CCB address
252 0312 2
253 0313 2 !+
254 0314 2 ! Find name of NAMELIST group and read records until it is found.
255 0315 2 !-
256 0316 2
257 0317 3 BEGIN
258 0318 3 LOCAL
259 0319 3 GROUP_NAME: REF VECTOR [, BYTE],
260 0320 3 RECORD_ADR: REF VECTOR [, BYTE];
261 0321 3
262 0322 3 GROUP_NAME = ..PARAM_BLOCK [NML$_LISTBLOCK];
263 0323 3 WHILE 1 DO
264 0324 4 BEGIN
265 0325 4
266 0326 4 BUILTIN
267 0327 4 LOCC;
268 0328 4
269 0329 4 !+
270 0330 4 ! Read a record and see if it has the right group name.
271 0331 4 ! STRING will be the remainder of the line if the search succeeds.
272 0332 4 !-
273 0333 4
274 0334 4 FOR$REC RSNO (); ! Get next record
275 0335 4 RECORD_ADR = .CCB [LUB$_BUF_PTR]; ! Record buffer address
276 0336 4
277 0337 4 !+
278 0338 4 ! Trim comment, if any, from record.
279 0339 4 !-
280 0340 4
281 0341 4 LOCC (%REF(%C'!'), %REF((.CCB [LUB$_BUF_END] - .RECORD_ADR)),
282 0342 4 RECORD_ADR [0];, CCB [LUB$_BUF_END]);
283 0343 4
284 0344 4 IF RECORD_ADR [2] LSSA .CCB [LUB$_BUF_END] ! Group name possible?
285 0345 4 THEN
286 0346 5 BEGIN
287 0347 5 IF .RECORD_ADR [1] EQLU %C'$' OR .RECORD_ADR [1] EQLU %C'&'
288 0348 5 THEN
289 0349 6 BEGIN
290 0350 6 LOCAL
291 0351 6 STRING_PTR,
292 0352 6 GROUPNAME_LEN;
```



```
293 0353 6      GROUPNAME_LEN = .GROUP_NAME [0];
294 0354 6      RECORD_ADR = RECORD_ADR [1];      ! Start with second char
295 0355 6      !+
296 0356 6      ! Set up STRINGPTR to be the remainder of the record after
297 0357 6      ! the first blank or tab, if any.
298 0358 6      !-
299 0359 6
300 0360 6      STRING_PTR = RECORD_ADR [1];
301 0361 6      WHILE (.STRING_PTR [SSA .CCB [LUB$A_BUF_END]]) DO
302 0362 7      BEGIN
303 0363 7      LITERAL
304 0364 7      K_TAB = 9; ! <TAB> character.
305 0365 7      IF CH$RCHAR (.STRING_PTR) EQLU %C' ' OR
306 0366 7      CH$RCHAR (.STRING_PTR) EQLU K_TAB
307 0367 7      THEN
308 0368 7      EXITLOOP;
309 0369 7      STRING_PTR = .STRING_PTR + 1;
310 0370 6      END;
311 0371 6
312 0372 6      PARAM_BLOCK [TPA$L_STRINGPTR] = .STRING_PTR;
313 0373 6      PARAM_BLOCK [TPA$L_STRINGCNT] = .CCB [LOB$A_BUF_END] -
314 0374 6      .STRING_PTR;
315 0375 6
316 0376 6      !+
317 0377 6      ! Compare the group name with the string after the $ or &
318 0378 6      !-
319 0379 6
320 0380 7      IF .GROUPNAME_LEN EQL ((.PARAM_BLOCK [TPA$L_STRINGPTR] - .RECORD_ADR) - 1)
321 0381 7      THEN IF (INCR POS FROM 1 TO .GROUPNAME_LEN DO
322 0382 8      BEGIN
323 0383 8      IF .GROUP_NAME [.POS] NEQ
324 0384 9      (
325 0385 9      IF (.RECORD_ADR [.POS] GEQ %C'a') AND
326 0386 10      (.RECORD_ADR [.POS] LEQ %C'z')
327 0387 9      THEN
328 0388 10      .RECORD_ADR [.POS] - (%C'a' - %C'A')
329 0389 9      ELSE
330 0390 9      .RECORD_ADR [.POS]
331 0391 9      )
332 0392 8      THEN
333 0393 8      EXITLOOP 0; ! Mismatch found
334 0394 6      END) NEQ 0
335 0395 6      THEN
336 0396 6      EXITLOOP;      ! Match found
337 0397 5      END;
338 0398 4      END;
339 0399 4
340 0400 4      !+
341 0401 4      ! Group name not found. See if the first non-blank is a '?',
342 0402 4      ! in which case we'll display the group we're looking for.
343 0403 4      !-
344 0404 4
345 0405 4      IF RECORD_ADR [1] LSSA .CCB [LUB$A_BUF_END]
346 0406 4      THEN
347 0407 5      BEGIN
348 0408 5      LOCAL
349 0409 5      CHARS_REMAINING,
```



```
350      0410      5      FOUND_CHAR: REF VECTOR [, WORD];
351      0411      5
352      0412      5      BUILTIN
353      0413      5      SKPC;
354      0414      5
355      0415      5      RECORD_ADR = RECORD_ADR [1];
356      0416      5      SKPC (%REF(%C' '), %REF((.CCB [LUB$A BUF_END] - .RECORD_ADR)),
357      0417      5      RECORD_ADR [0]; CHARS_REMAINING, FOUND_CHAR);
358      0418      5
359      0419      5      !+
360      0420      5      FOUND_CHAR now contains address of first non-blank character.
361      0421      5      See if it is a "?" or "=?".
362      0422      5      !-
363      0423      5
364      0424      5      IF .CHARS_REMAINING NEQ 0
365      0425      5      THEN
366      0426      6      BEGIN
367      0427      6      BIND
368      0428      6      FAB = CCB: REF $FOR$FAB CCB STRUCT,
369      0429      6      FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
370      0430      6
371      0431      6      IF .FAB_DEV [DEV$V_TRM] AND .FAB [FAB$V_PUT]
372      0432      6      THEN
373      0433      7      BEGIN
374      0434      7      IF CH$RCHAR (.FOUND_CHAR) EQL %C'?'
375      0435      7      THEN
376      0436      8      BEGIN
377      0437      8      FOR$$REC WSNO (); ! Start an output record
378      0438      8      FOR$$DO_NML_OUTPUT (1); ! Dump names only
379      0439      8      END
380      0440      7      ELSE IF .CHARS_REMAINING GTRU 1
381      0441      7      THEN
382      0442      8      BEGIN
383      0443      8      IF .FOUND_CHAR [0] EQL %ASCII'='
384      0444      8      THEN
385      0445      9      BEGIN
386      0446      9      FOR$$REC WSNO (); ! Start an output record
387      0447      9      FOR$$DO_NML_OUTPUT (0); ! Dump values too
388      0448      8      END;
389      0449      7      END;
390      0450      6      END;
391      0451      5      END;
392      0452      4      END;
393      0453      4
394      0454      3      END; ! Loop forever until EOF or match
395      0455      3
396      0456      2      END;
397      0457      2
398      0458      2
399      0459      2      !+
400      0460      2      At this point, STRINGPTR and STRINGCNT reflect the rest of the record
401      0461      2      after the group name (possibly empty). We now call LIB$TPARSE to
402      0462      2      analyze and execute any assignments contained in the NAMELIST input.
403      0463      2      The parse will terminate when a end-of-block or error is found.
404      0464      2      !-
405      0465      2      IF NOT LIB$TPARSE (PARAM_BLOCK, FOR$$A_NMLSTATE, FOR$$A_NMLKEYWD)
406      0466      2      THEN
```


51	6045	51	08	6042	04	11	000A1	BRB	7\$	:	0390
				00	9A	000A3	6\$:	MOVZBL	(POS)[RECORD_ADR], R1	:	0384
				04	ED	000A7	7\$:	CMPZV	#0, #8, (POST)[GROUP_NAME], R1	:	
				50	13	000AD		BEQL	8\$	:	0393
				07	D4	000AF		CLRL	R0	:	
				54	11	000B1		BRB	9\$	:	
	D5	50		01	F3	000B3	8\$:	AOBLEQ	GROUPNAME_LEN, POS, 5\$	:	0381
		50		4C	CE	000B7		MNEGL	#1, R0	:	
		50		50	12	000BA	9\$:	BNEQ	14\$	:	0394
		53	01	A2	9E	000BC	10\$:	MOVAB	1(R2), R0	:	0405
				50	D1	000C0		CMPL	R0, R3	:	
				40	1E	000C3		BGEQU	13\$	:	
				52	D6	000C5		INCL	RECORD_ADR	:	0415
	50	53		52	C3	000C7		SUBL3	RECORD_ADR, R3, R0	:	0416
	62	50		20	3B	000CB		SKPC	#32, R0, (RECORD_ADR)	:	0417
		54		50	D0	000CF		MOVL	R0, R4	:	
		53		51	D0	000D2		MOVL	R1, R3	:	
				54	D5	000D5		TSTL	CHARS_REMAINING	:	0424
				2C	13	000D7		BEQL	13\$	:	
	26	0084	CB	02	E1	000D9		BBC	#2, 132(CCB), 13\$	:	0431
			22	AB	E9	000DF		BLBC	90(CCB), 13\$	:	
			3F	63	91	000E3		CMPB	(FOUND_CHAR), #63	:	0434
				06	12	000E6		BNEQ	11\$	:	
				66	16	000E8		JSB	FOR\$\$REC_WSNO	:	0437
				01	DD	000EA		PUSHL	#1	:	0438
				10	11	000EC		BRB	12\$	:	
		01		54	D1	000EE	11\$:	CMPL	CHARS_REMAINING, #1	:	0440
				12	1B	000F1		BLEQU	13\$	:	
	3F3D	8F		63	B1	000F3		CMPW	(FOUND_CHAR), #16189	:	0443
				0B	12	000F8		BNEQ	13\$	:	
				66	16	000FA		JSB	FOR\$\$REC_WSNO	:	0446
				7E	D4	000FC		CLRL	-(SP)	:	0447
	00000000G	00		01	FB	000FE	12\$:	CALLS	#1, FOR\$\$DO_NML_OUTPUT	:	
				FF1F	31	00105	13\$:	BRW	1\$	:	0431
				00	9F	00108	14\$:	PUSHAB	FOR\$\$A_NMLKEYWD	:	0465
				00	9F	0010E		PUSHAB	FOR\$\$A_NMLSTATE	:	
			08	AE	9F	00114		PUSHAB	PARAM_BLOCK	:	
	00000000G	00		03	FB	00117		CALLS	#3, LIB\$TPARSE	:	
		10		50	E8	0011E		BLBS	R0, 15\$	:	
			10	AE	9F	00121		PUSHAB	PARAM_BLOCK+16	:	0467
			0018808C	8F	DD	00124		PUSHL	#1605772	:	
	00000000G	00		02	FB	0012A		CALLS	#2, FOR\$\$SIGNAL_STO	:	
				04	00	00131	15\$:	RET		:	0470

; Routine Size: 306 bytes. Routine Base: _FOR\$CODE + 0006

; 411 0471 1 !<BLF/PAGE>

FOR\$\$UDF_RN
1-005

NAMLIST input UDF Level
FOR\$\$UDF_RN9 - Terminate NAMLIST input

G 10
16-Sep-1984 00:49:15
14-Sep-1984 12:32:51

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

Page 12
(7)

```
: 413 0472 1 %SBTTL 'FOR$$UDF_RN9 - Terminate NAMLIST input'
: 414 0473 1 GLOBAL ROUTINE FOR$$UDF_RN9 ! Terminate NAMLIST input
: 415 0474 1 : JSB_UDF9 NOVALUE =
: 416 0475 1
: 417 0476 1 ++
: 418 0477 1 FUNCTIONAL DESCRIPTION:
: 419 0478 1
: 420 0479 1 Terminate NAMLIST input. This procedure is necessary because
: 421 0480 1 FOR$IO_END dispatches to a UDF9 routine for all statement types.
: 422 0481 1
: 423 0482 1 CALLING SEQUENCE:
: 424 0483 1
: 425 0484 1 JSB FOR$$UDF_RN9
: 426 0485 1
: 427 0486 1 FORMAL PARAMETERS:
: 428 0487 1
: 429 0488 1 NONE
: 430 0489 1
: 431 0490 1 IMPLICIT INPUTS:
: 432 0491 1
: 433 0492 1 NONE
: 434 0493 1
: 435 0494 1 IMPLICIT OUTPUTS:
: 436 0495 1
: 437 0496 1 NONE
: 438 0497 1
: 439 0498 1 COMPLETION STATUS:
: 440 0499 1
: 441 0500 1 NONE
: 442 0501 1
: 443 0502 1 SIDE EFFECTS:
: 444 0503 1
: 445 0504 1 NONE
: 446 0505 1
: 447 0506 1 --
: 448 0507 1
: 449 0508 2 BEGIN
: 450 0509 2
: 451 0510 2 RETURN;
: 452 0511 2
: 453 0512 1 END; ! End of routine FOR$$UDF_RN9
```

05 00000 FOR\$\$UDF_RN9::
RSB

; 0512

; Routine Size: 1 bytes, Routine Base: _FOR\$CODE + 0138

; 454 0513 1 !<BLF/PAGE>

FOR\$\$UDF_RN
1-005

NAMLIST input UDF level
FOR\$\$UDF_RN9 - Terminate NAMLIST input

H 10
16-Sep-1984 00:49:15 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:32:51 [FORRTL.SRC]FORUDFRN.B32;1

: 456 0514 1 END
: 457 0515 1
: 458 0516 0 ELUDOM

! End of module FOR\$\$UDF_RN

PSECT SUMMARY

Name Bytes Attributes
_FOR\$CODE 313 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	8	0	581	00:01.0
-\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	207	29	52	00:00.6
-\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1
-\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	0	0	14	00:00.2

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:FORUDFRN/OBJ=OBJ\$:FORUDFRN MSRC\$:FORUDFRN/UPDATE=(ENH\$:FORUDFRN)
: Size: 313 code + 0 data bytes
: Run Time: 00:10.7
: Elapsed Time: 00:37.3
: Lines/CPU Min: 2907
: Lexemes/CPU-Min: 11836
: Memory Used: 154 pages
: Compilation Complete

0184 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY