



```

FFFFFFFFF 000000 RRRRRRRR SSSSSSSS IIIIII GGGGGGGG NN NN AAAAAA LL
FFFFFFFFF 000000 RRRRRRRR SSSSSSSS IIIIII GGGCGGGG NN NN AAAAAA LL
FF 00 00 RR RR SS II GG NN NN AA AA LL
FF 00 00 RR RR SS II GG NN NN AA AA LL
FF 00 00 RR RR SS II GG NN NN AA AA LL
FF 00 00 RR RR SS II GG NN NN AA AA LL
FFFFFFFF 00 00 RRRRRRRR SSSSSS II GG NN NN AA AA LL
FFFFFFFF 00 00 RRRRRRRR SSSSSS II GG NN NN AA AA LL
FF 00 00 RR RR SS II GG GGGGGG NN NN AAAAAAAA LL
FF 00 00 RR RR SS II GG GGGGGG NN NN AAAAAAAA LL
FF 00 00 RR RR SS II GG GG NN NN AA AA LL
FF 00 00 RR RR SS II GG GG NN NN AA AA LL
FF 000000 RR RR SSSSSSSS IIIIII GGGGGG NN NN AA AA LL
FF 000000 RR RR SSSSSSSS IIIIII GGGGGG NN NN AA AA LLLLLLLLLL

```

```

LL IIIIII SSSSSSSS
LL IIIIII 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 IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```

```

1 0001 0 MODULE FOR$$SIGNAL (%TITLE'FORTRAN SIGNAL, SIGNAL_STOP and SIG_NO_LUB'
2 0002 0 IDENT = '1-007' ! File: FORSIGNAL.B32 Edit: SBL1007
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 |
7 0007 1 |*****
8 0008 1 |*
9 0009 1 |* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 |* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 |* ALL RIGHTS RESERVED. *
12 0012 1 |*
13 0013 1 |* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 |* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 |* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 |* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 |* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 |* TRANSFERRED. *
19 0019 1 |*
20 0020 1 |* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 |* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 |* CORPORATION. *
23 0023 1 |*
24 0024 1 |* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 |* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 |*
27 0027 1 |*
28 0028 1 |*****
29 0029 1 |
30 0030 1 |
31 0031 1 |++
32 0032 1 | FACILITY: FORTRAN Support Library
33 0033 1 |
34 0034 1 | ABSTRACT:
35 0035 1 |
36 0036 1 | FORTRAN support routines to convert FORTRAN error code
37 0037 1 | to 32-bit VAX error code, and SIGNAL or SIGNAL_STOP
38 0038 1 | extra information in format compatible for SYS$PUT_MESSAGE:
39 0039 1 |
40 0040 1 | ENVIRONMENT: User Mode - AST re-entrant
41 0041 1 | Note: this module is both shared and non-shared.
42 0042 1 | If compatibility routine calls it, a non-shared copy is included.
43 0043 1 | Hence, JSB to FOR$$CB_GET instead of accessing OTS$$A_CUR_LUB directly.
44 0044 1 |
45 0045 1 | AUTHOR: Thomas N. Hastings, CREATION DATE: 8-Aug-1977
46 0046 1 |
47 0047 1 | MODIFIED BY:
48 0048 1 |
49 0049 1 | Thomas N. Hastings, 8-Aug-1977: VERSION 0
50 0050 1 | Steven B. Lionel, VAX/VMS V2.0
51 0051 1 | [Previous edit history removed. SBL 10-Nov-1980]
52 0052 1 | 1-001 - Update version number and copyright notice. JBS 16-NOV-78
53 0053 1 | 1-002 - Change LUB$B_LUN to LUB$W_LUN. JBS 05-DEC-78
54 0054 1 | 1-003 - Change REQUIRE file names from FOR... to OTS... JBS 06-DEC-78
55 0055 1 | 1-004 - Get filename from FAB if all else fails. SBL 29-Aug-1979
56 0056 1 | 1-005 - Add optional FAB argument to FOR$$SIG_NO_LUB. SBL 7-OCT-1979
57 0057 1 | 1-006 - Allow extra FAO arguments and conditions to be passed to

```

FOR\$\$\$IGNAL  
1-007

FORTTRAN SIGNAL, SIGNAL\_STOP and SIG\_NO\_LUB

F 16  
16-Sep-1984 00:44.51  
14-Sep-1984 12:32:44

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FOR\$\$\$IGNAL.B32;1

```
: 58      0058 1 !          FOR$$$IGNAL and FOR$$$IGNAL_STO. Remove debugging macros, no longer  
: 59      0059 1 !          used. SBL 10-Nov-1980  
: 60      0060 1 !          *** First post-V3.0 edit ***  
: 61      0061 1 !          1-007 - Use prologue file. SBL 20-Jan-1983  
: 62      0062 1 !          --
```



FORSSIGNAL  
1-007

FORTRAN SIGNAL, SIGNAL\_STOP and SIG\_NO\_LUB

H 16  
16-Sep-1984 00:44:51  
14-Sep-1984 12:32:44

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

Page 4  
(2)

```
: 121      0185 1 EXTERNAL LITERAL
: 122      0186 1
: 123      0187 1      OTSS_FATINTERR: UNSIGNED (%BPVAL),      ! Condition value FATAL INTERNAL ERROR IN RUN-TIME LIBRARY
: 124      0188 1      O*SS_INTDATCOR: UNSIGNED (%BPVAL);      ! Condition value INTERNAL DATA CORRUPTED IN RUN-TIME LIBRAR
```

```
126 0189 1 GLOBAL ROUTINE FOR$$SIGNAL      ! SIGNAL FORTRAN error and continue
127 0190 1      :NOVALUE =                    . No value returned.
128 0191 1
129 0192 1  +-+
130 0193 1  FUNCTIONAL DESCRIPTION:
131 0194 1      Signals a FORTRAN-specific error whose 32-bit condition code or
132 0195 1      small-integer error number is the first argument.  If other arguments
133 0196 1      are present, they represent extra FAO arguments for the first
134 0197 1      condition and/or secondary conditions to be signalled.  See DO_SIGNAL
135 0198 1      for more information
136 0199 1
137 0200 1  CALLING SEQUENCE:
138 0201 1
139 0202 1      CALL FOR$$SIGNAL (fort_err_no.rc.v [,fao_args_0.rz.v, ...]
140 0203 1      [,secondary_msg.rc.v [,sec_fao_cnt.rl.v[, sec_fao_args.rz.v,...]])
141 0204 1
142 0205 1  FORMAL PARAMETERS:
143 0206 1
144 0207 1      fort_err_no      - A 32-bit FOR$ code or the small integer which is the
145 0208 1      error number part of a FOR$ code.
146 0209 1      fao_args_0      - FAO arguments for this message.  The three FAO arguments
147 0210 1      unit number, filename and user PC are always used; if
148 0211 1      fao_args_0 are specified, they come before the default
149 0212 1      arguments.
150 0213 1      secondary_msg  - Secondary message to be signalled.  MUST be a 32-bit code.
151 0214 1      sec_fao_cnt    - FAO count for secondary message
152 0215 1      sec_fao_args   - FAO arguments for secondary message
153 0216 1
154 0217 1  IMPLICIT INPUTS:
155 0218 1
156 0219 1      See DO_SIGNAL
157 0220 1
158 0221 1  IMPLICIT OUTPUTS:
159 0222 1
160 0223 1      See DO_SIGNAL
161 0224 1
162 0225 1  COMPLETION CODES:
163 0226 1
164 0227 1      NONE
165 0228 1
166 0229 1  SIDE EFFECTS:
167 0230 1
168 0231 1      Converts FORTRAN error code to 32-bit VAX-11 error code and SIGNALs.
169 0232 1      Saves error info in FOR$ERRSNS OWN storage.
170 0233 1  --
171 0234 1
172 0235 2  BEGIN
173 0236 2
174 0237 2  BUILTIN
175 0238 2  AP;
176 0239 2
177 0240 2  DO_SIGNAL (.AP, LIB$$SIGNAL);
178 0241 2
179 0242 2  RETURN
180 0243 1  END;
```

FOR\$\$SIGNAL  
1-007

FORTTRAN SIGNAL, SIGNAL\_STOP and SIG\_NO\_LUB

J 16  
16-Sep-1984 00:44:51  
14-Sep-1984 12:32:44

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

Page 6  
(3)

.TITLE FOR\$\$SIGNAL FORTTRAN SIGNAL, SIGNAL\_STOP and SIG  
\_NO\_LUB

.IDENT \1-007\

.EXTRN FOR\$\$CB GET, FOR\$\$ERRSNS\_SAV

.EXTRN LIB\$\$SIGNAL, LIB\$\$STOP

.EXTRN OT\$\$\_FATINTERR, OT\$\$\_INTDATCOR

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

.ENTRY FOR\$\$SIGNAL, Save nothing

PUSHAB LIB\$\$SIGNAL

PUSHL AP

CALLS #2, DO\_SIGNAL

RET

: 0189

: 0240

: 0243

			0000	00000
	00000000G	00	9F	00002
		5C	DD	00008
0000V	CF	02	FB	0000A
		04	00	0000F

: Routine Size: 16 bytes, Routine Base: \_FOR\$CODE + 0000



```

: 182      0244 1 GLOBAL ROUTINE FOR$$SIGNAL_STO  ! SIGNAL_STOP FORTRAN error and STOP
: 183      0245 1      :NOVALUE =                ! No value returned.
: 184      0246 1      !++
: 185      0247 1      FUNCTIONAL DESCRIPTION:
: 186      0248 1
: 187      0249 1      Convert FORTRAN error code number to 32-bit VAX-11 error code.
: 188      0250 1      See description for FOR$$SIGNAL above which is identical,
: 189      0251 1      except that FOR$$SIGNAL_STO calls LIB$STOP instead of LIB$SIGNAL.
: 190      0252 1      !--
: 191      0253 1
: 192      0254 2      BEGIN
: 193      0255 2
: 194      0256 2      BUILTIN
: 195      0257 2      AP;
: 196      0258 2
: 197      0259 2      DO_SIGNAL (.AP, LIB$STOP);
: 198      0260 2
: 199      0261 2      RETURN
: 200      0262 1      END;

```

```

                                0000 00C00
                                00000000G 00 9F 00002
                                5C DD 00008
                                0000V CF 02 FB 0000A
                                04 0000F
                                .ENTRY FOR$$SIGNAL_STO, Save nothing
                                PUSHAB LIB$STOP
                                PUSHL AP
                                CALLS #2, DO_SIGNAL
                                RET
: 0244
: 0259
:
: 0262

```

: Routine Size: 16 bytes, Routine Base: \_FOR\$CODE + 0010

```

: 202      0263 1 GLOBAL ROUTINE FOR$$SIG_FATINT      ! SIGNAL_STOP OTSS_FATINTERR and STOP
: 203      0264 1      :NOVALUE =                  ! No value returned.
: 204      0265 1      !++
: 205      0266 1      ! FUNCTIONAL DESCRIPTION:
: 206      0267 1      !
: 207      0268 1      !     SIGNAL_STOP OTSS_FATINTERR = FATAL INTERNAL ERROR IN RUN-TIME LIBRARY.
: 208      0269 1      !     Note: The current LUB (if any) is ignored and no UNIT is printed.
: 209      0270 1      !     !--
: 210      0271 1
: 211      0272 2      BEGIN
: 212      0273 2      FOR$$SIG_NO_LUB (OTSS_FATINTERR);
: 213      0274 2      RETURN
: 214      0275 1      END;

```

```

                                0000 0000      .ENTRY FOR$$SIG_FATINT, Save nothing      : 0263
                                8F DD 00002     PUSHL #OTSS_FATINTERR                    : 0273
0000V CF 00000000G 01 FB 00008             CALLS #1, FOR$$SIG_NO_LUB                :
                                04 0000D             RET                               : 0275

```

; Routine Size: 14 bytes, Routine Base: \_FOR\$CODE + 0020

```

: 216      0276 1 GLOBAL ROUTINE FOR$$SIG_DATCOR      ! SIGNAL_STOP OTSS_INTDATCOR and STOP
: 217      0277 1      :NOVALUE =                  ! No value returned.
: 218      0278 1      !++
: 219      0279 1      ! FUNCTIONAL DESCRIPTION:
: 220      0280 1      !
: 221      0281 1      !     SIGNAL_STOP OTSS_INTDATCOR = INTERNAL DATA CORRUPTED IN RUN-TIME LIBRARY.
: 222      0282 1      !     Note: The current LUB (if any) is ignored and no UNIT is printed.
: 223      0283 1      ! --
: 224      0284 1
: 225      0285 2      BEGIN
: 226      0286 2      FOR$$SIG_NO_LUB (OTSS_INTDATCOR);
: 227      0287 2      RETURN
: 228      0288 1      END;

```

```

                                0000 0000      .ENTRY FOR$$SIG_DATCOR, Save nothing      : 0276
                                8F DD 0002      PUSHL #OTSS_INTDATCOR                    : 0286
0000V CF 00000000G 01 FB 0008      CALLS #1, FOR$$SIG_NO_LUB                    :
                                04 000D      RET                                          : 0288

```

; Routine Size: 14 bytes, Routine Base: \_FOR\$CODE + 002E

```
230 0289 1 ROUTINE DO SIGNAL (      ! Internal routine to do work for FOR$$SIGNAL and FOR$$SIGNAL_STO
231 0290 1     SIGNAL_LIST_ARG,      ! List of arguments to signal routine
232 0291 1     SIGNAL_ROUTINE)      ! adr. of LIB$$SIGNAL or LIB$$STOP
233 0292 1     : NOVALUE =           ! No value returned
234 0293 1
235 0294 1 ++
236 0295 1     FUNCTIONAL DESCRIPTION:
237 0296 1         Converts error code number to 32-bit VAX-11 error code.
238 0297 1         See description of FOR$$SIGNAL above.
239 0298 1
240 0299 1     FORMAL PARAMETERS:
241 0300 1
242 0301 1     SIGNAL_LIST_ARG      Contents of AP at time of call to FOR$$SIGNAL or
243 0302 1     FOR$$SIGNAL_STO
244 0303 1     SIGNAL_ROUTINE          Adr. of LIB$$SIGNAL or LIB$$STOP
245 0304 1
246 0305 1     IMPLICIT INPUTS:
247 0306 1
248 0307 1     OT$$A_CUR_LUB          Adr. of current LUB/ISB/RAB
249 0308 1     Obtained by JSB to FOR$$CB_GET.
250 0309 1     {FAB,RAB}$L_STS       RMS error status
251 0310 1     {FAB,RAB}$L_STV       RMS error value or operating system error code
252 0311 1
253 0312 1     IMPLICIT OUTPUTS:
254 0313 1
255 0314 1     {FAB,RAB}$L_STS       RMS error status - set to 0
256 0315 1     {FAB,RAB}$L_STV       RMS error value or operating system error code - set to 0
257 0316 1     FORTRAN error #, RMS STS, RMS STV, logical unit number saved in
258 0317 1     OWN storage in FOR$ERRSNS module for later
259 0318 1     call by user to ERRSNS.
260 0319 1
261 0320 1     COMPLETION CODES:
262 0321 1
263 0322 1     NONE
264 0323 1
265 0324 1     SIDE EFFECTS:
266 0325 1
267 0326 1     Converts FORTRAN error code to 32-bit VAX-11 error code and SIGNALS.
268 0327 1     Saves error info in FOR$ERRSNS OWN storage.
269 0328 1     --
270 0329 1
```

```

272 0330 2 BEGIN
273 0331 2
274 0332 2 GLOBAL REGISTER
275 0333 2 CCB = K_CCB_REG: REF $FOR$CCB_DECL;
276 0334 2
277 0335 2 LOCAL
278 0336 2 FILE_NAME_DSC: DSC$DESCRIPTOR, ! File name descriptor for resultant file name
279 0337 2 RABORFAB: REF BLOCK[, BYTE],
280 0338 2 STS, ! RMS RAB or FAB error status
281 0339 2 STV, ! RMS RAB or FAB error status
282 0340 2 GETMSG_VALS: VECTOR [4, BYTE], ! Returned values from $GETMSG
283 0341 2 SIGNAL_LIST: VECTOR [20, LONG], ! Argument list to LIB$$SIGNAL/LIB$$STOP
284 0342 2 LIST_PTR: REF VECTOR [, LONG], ! Pointer into signal list
285 0343 2 ARGS_PTR: REF VECTOR [, LONG], ! pointer into SIGNAL_LIST_ARG
286 0344 2 ARG [LIST_END, ! Address of argument list end
287 0345 2 COND_VAL: BLOCK [4, BYTE]; ! 32-bit VAX-11 error code
288 0346 2
289 0347 2 MAP
290 0348 2 SIGNAL_LIST_ARG: REF VECTOR [, LONG];
291 0349 2
292 0350 2 BUILTIN
293 0351 2 CALLG;
294 0352 2
295 0353 2 FOR$$CB_GET (); ! Set CCB to adr. of current LUB/ISB/RAB
296 0354 2
297 0355 2
298 0356 2 +
299 0357 2 Convert FORTRAN error code to 32-bit VAX-11 error code.
300 0358 2 Conversion is done by copying FORTRAN error number to code field,
301 0359 2 setting the severity code to SEVERE,
302 0360 2 for all errors except FOR$_OUTCONERR (63='OUTPUT CONVERSION ERROR')
303 0361 2 which is set to ERROR instead so that image will continue
304 0362 2 by default since output field is flagged with **$.
305 0363 2 All other continuable errors are signaled SEVERE so that user
306 0364 2 must take overt action in order to continue past the error.
307 0365 2 setting the facility code to FOR$K FAC NO,
308 0366 2 and setting the facility specific bit (ST$$V_FAC_SP).
309 0367 2 -
310 0368 2 COND_VAL = COND_VALUE (.SIGNAL_LIST_ARG [1]);
311 0369 2
312 0370 2
313 0371 2 +
314 0372 2 Call $GETMSG to see how many FAO parameters it takes.
315 0373 2 -
316 0374 2 BEGIN
317 0375 2 LOCAL
318 0376 2 DSC: VECTOR [2, LONG],
319 0377 2 LEN;
320 0378 2 DSC [0] = 0; ! Null string descriptor
321 0379 2 DSC [1] = LEN;
322 0380 2 $GETMSG (
323 0381 2 MSGID = .COND_VAL,
324 0382 2 MSGLEN = LEN,
325 0383 2 BUFADR = DSC,
326 0384 2 FLAGS = 0,
327 0385 2 OUTADR = GETMSG_VALS);
328 0386 2 END;

```

```
329 0387 2
330 0388 2
331 0389 2  !+ Compute total number of signal arguments.
332 0390 2  !-
333 0391 2
334 0392 2 SIGNAL_LIST [0] = (.SIGNAL_LIST_ARG [0])<0,8,0> + 6;
335 0393 2 ARG_LIST_END = SIGNAL_LIST_ARG [0] + ((.SIGNAL_LIST_ARG [0])<0,8,0> * %UPVAL);
336 0394 2
337 0395 2
338 0396 2  !+ Fill in primary condition message.
339 0397 2  !-
340 0398 2
341 0399 2 SIGNAL_LIST [1] = .COND_VAL;
342 0400 2 SIGNAL_LIST [2] = .GETMSG_VALS [1]; ! Number of FAO parameters
343 0401 2
344 0402 2 LIST_PTR = SIGNAL_LIST [3];
345 0403 2 ARGS_PTR = SIGNAL_LIST_ARG [2];
346 0404 2
347 0405 2  !+
348 0406 2  !- Copy extra FAO arguments, if any.
349 0407 2
350 0408 2
351 0409 2 INCR I FROM 4 TO .SIGNAL_LIST [2] DO
352 0410 2     COPY_LONG_A (ARGS_PTR, LIST_PTR);
353 0411 2
354 0412 2  !+
355 0413 2  !- Get RMS error status from RAB or if not error there from FAB (if any).
356 0414 2  !- Then set error status longwords to 0 so will not be found again.
357 0415 2  !- Note: this code depends on the fact that RAB$L_STS/STV have the same offsets
358 0416 2  !- as RAB$L_STS/STV.
359 0417 2  !-
360 0418 2
361 0419 2 STS = 0; ! Set initial values
362 0420 2 STV = 0;
363 0421 2 IF .CCB [LUB$W_LUN] NEQU LUB$K_LUN_ENCD ! Not ENCODE/DECODE/internal?
364 0422 2 THEN
365 0423 2     BEGIN
366 0424 2     RABORFAB = .CCB;
367 0425 2     IF (.CCB[RAB$L_STS] OR .CCB[RAB$L_STS] EQL 0)
368 0426 2     THEN
369 0427 2     RABORFAB = .CCB[RAB$L_FAB];
370 0428 2
371 0429 2     IF NOT .RABORFAB[RAB$L_STS]
372 0430 2     THEN
373 0431 2     BEGIN
374 0432 2     STS = .RABORFAB[RAB$L_STS];
375 0433 2     STV = .RABORFAB[RAB$L_STV];
376 0434 2     END;
377 0435 2
378 0436 2     RABORFAB[RAB$L_STS] = 0;
379 0437 2     RABORFAB[RAB$L_STV] = 0;
380 0438 2     END;
381 0439 2
382 0440 2  !+
383 0441 2  !- Save FORTRAN error number, RMS STS, RMS STV, logical unit number and VAX-11 condition value
384 0442 2
385 0443 2
```

```

386 0444 2 FOR$$ERRSNS_SAV (.COND_VAL [ST$$V_CODE], .STS, .STV, .CCB[LUB$$W_LUN], .COND_VAL);
387 0445 2
388 0446 2
389 0447 2 !+
390 0448 2 ! Set up resultant file name descriptor that gets put in signal arg list.
391 0449 2 ! Note that this points at the FAB's FNM until the file is opened.
392 0450 2
393 0451 2 FILE_NAME_DSC[DSC$$W_LENGTH] = .CCB[LUB$$B_RSL];
394 0452 2 FILE_NAME_DSC[DSC$$B_DTYPE] = FILE_NAME_DSC[DSC$$B_CLASS] = 0;
395 0453 2 FILE_NAME_DSC[DSC$$A_POINTER] =
396 0454 2 ? IF .CCB[LUB$$B_RSL] EQLU 0
397 0455 2 THEN
398 0456 2 0
399 0457 2 ELSE
400 0458 2 .CCB[LUB$$A_RSN]);
401 0459 2
402 0460 2 !+
403 0461 2 ! Insert the three default FAO arguments plus the STS and STV.
404 0462 2
405 0463 2
406 0464 2 LIST_PTR [0] = .CCB [LUB$$W_LUN];
407 0465 2 LIST_PTR [1] = FILE_NAME_DSC;
408 0466 2 LIST_PTR [2] = 0; ! For user PC
409 0467 2 LIST_PTR [3] = .STS;
410 0468 2 LIST_PTR [4] = .STV;
411 0469 2 LIST_PTR = LIST_PTR [5];
412 0470 2
413 0471 2 WHILE .ARGS_PTR LEQ .ARG_LIST_END DO
414 0472 2 COPY_LONG_A (ARGS_PTR, LIST_PTR);
415 0473 2
416 0474 2 !+
417 0475 2 ! Call LIB$STOP to STOP the error or LIB$SIGNAL to SIGNAL the error.
418 0476 2
419 0477 2
420 0478 2 CALLG (SIGNAL_LIST, .SIGNAL_ROUTINE);
421 0479 2
422 0480 2 !+
423 0481 2 ! Return
424 0482 2
425 0483 2
426 0484 2 RETURN
427 0485 1 END; ! End of FOR$$SIGNAL_STO routine

```

.EXTRN SYS\$GETMSG

				08FC 0000 DO_SIGNAL:			
	SE	98	AE 9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R11	:	0289
		00000000G	00 16 00006	MOVAB	-104(SP), SP	:	
	52	04	AC D0 0000C	JSB	FOR\$\$CB_GET	:	0353
		04	A2 DD 00010	MOVL	SIGNAL_LIST_ARG, R2	:	0368
	0000V	CF	01 FB 00013	PUSHL	4(R2)	:	
		54	50 D0 00018	CALLS	#1, COND_VALUE	:	
			50 D0 00018	MOVL	R0, COND_VAL	:	
		08	AE D4 0001B	CLRL	DSC	:	0378
	0C	AE	6E 9E 0001E	MOVAB	LEN, DSC+4	:	0379

		04	AE	9F	00022		PUSHAB	GETMSG_VALS	0385	
			7E	D4	00025		CLRL	-(SP)		
		10	AE	9F	00027		PUSHAB	DSC		
		0C	AE	9F	0002A		PUSHAB	LEN		
			54	DD	0002D		PUSHL	COND_VAL		
00000000G	00		05	FB	0002F		CALLS	#5, SYS\$GETMSG		
	50		62	D0	00036		MOVL	(R2), R0	0392	
	10	AE	06	A0	9E	00039	MOVAB	6(R0), SIGNAL_LIST		
	57		8240	DE	0003E		MOVAL	(R2)+[R0], ARG_LIST_END	0393	
	14	AE		54	D0	00042	MOVL	COND_VAL, SIGNAL_LIST+4	0399	
	18	AE	05	AE	9A	00046	MOVZBL	GETMSG_VALS+1, SIGNAL_LIST+8	0400	
	53		1C	AE	9E	0004B	MOVAB	SIGNAL_LIST+12, LIST_PTR	0402	
	52			04	C0	0004F	ADDL2	#4, ARGS_PTR	0403	
	50			03	D0	00052	MOVL	#3, I	0409	
				03	11	00055	BRB	2\$		
	83			82	D0	00057	1\$:	MOVL	(ARGS_PTR)+, (LIST_PTR)+	0410
F8	50		18	AE	F3	0005A	2\$:	AOBLEQ	SIGNAL_LIST+8, I, T\$	
				55	7C	0005F		CLRQ	STV	0420
	FFF8	8F	C6	AB	B1	00061		CMPW	-58(CCB), #-5	0421
				1F	13	00067		BEQL	6\$	
	50			5B	D0	00069		MOVL	CCB, RABORFAB	0424
	05		08	AB	E8	0006C		BLBS	8(CCB), 3\$	0425
			08	AB	D5	00070		TSTL	8(CCB)	
				04	12	00073		BNEQ	4\$	
	50		3C	AB	D0	00075	3\$:	MOVL	60(CCB), RABORFAB	0427
	08		08	A0	E8	00079	4\$:	BLBS	8(RABORFAB), 5\$	0429
	56		08	A0	D0	0007D		MOVL	8(RABORFAB), STS	0432
	55		0C	A0	D0	00081		MOVL	12(RABORFAB), STV	0433
			08	A0	7C	00085	5\$:	CLRQ	8(RABORFAB)	0436
				54	DD	00088	6\$:	PUSHL	COND_VAL	0444
	7E		C6	AB	32	0008A		CVTWL	-58(CCB), -(SP)	
				55	DD	0008E		PUSHL	STV	
				56	DD	00090		PUSHL	STS	
7E	54			03	EF	00092		EXTZV	#3, #12, COND_VAL, -(SP)	
	00000000G	0C		05	FB	00097		CALLS	#5, FOR\$\$ERRSNS SAV	
	60	AE	F7	AB	9B	0009E		MOVZBW	-9(CCB), FILE_NAME_DSC	0451
			62	AE	B4	000A3		CLRW	FILE_NAME_DSC+2	0452
			F7	AB	95	000A6		TSTB	-9(CCB)	0454
				04	12	000A9		BNEQ	7\$	
				50	D4	000AB		CLRL	R0	
				04	11	000AD		BRB	8\$	
	50		F8	AB	D0	000AF	7\$:	MOVL	-8(CCB), R0	0458
	64	AE		50	D0	000B3	8\$:	MOVL	R0, FILE_NAME_DSC+4	0454
	83		C6	AB	32	000B7		CVTWL	-58(CCB), (LIST_PTR)+	0464
	83		60	AE	9E	000BB		MOVAB	FILE_NAME_DSC, (LIST_PTR)+	0465
				83	D4	000BF		CLRL	(LIST_PTR)+	0466
	83			56	D0	000C1		MOVL	STS, (LIST_PTR)+	0467
	83			55	D0	000C4		MOVL	STV, (LIST_PTR)+	0468
	57			52	D1	000C7	9\$:	CMP	ARGS_PTR, ARG_LIST_END	0471
				05	14	000CA		BGTR	10\$	
	83			82	D0	000CC		MOVL	(ARGS_PTR)+, (LIST_PTR)+	0472
				F6	11	000CF		BRB	9\$	
	08	BC	10	AE	FA	000D1	10\$:	CALLG	SIGNAL_LIST, @SIGNAL_ROUTINE	0478
				04	000D6			RET		0485

; Routine Size: 215 bytes, Routine Base: \_FOR\$CODE + 003C



FORSSIGNAL  
1-007

FORTRAN SIGNAL, SIGNAL\_STOP and SIG\_NO\_LUB

C 1  
16-Sep-1984 00:44:51  
14-Sep-1984 12:32:44

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

Page 15  
(8)

FO  
1-

```
429 0486 1 GLOBAL ROUTINE FOR$$SIG_NO_LUB ( ! SIGNAL_STOP FORTRAN error and STOP
430 0487 1 FORT_ERR_NO, ! FORTRAN error code 0:120 or 32-bit cond value
431 0488 1 ! VAX-11 error code
432 0489 1 FORT_LUN, ! Optional FORTRAN logical unit number
433 0490 1 FAB) ! Optional FAB address
434 0491 1 :NOVALUE = ! No value returned.
435 0492 1
436 0493 1 ++
437 0494 1 FUNCTIONAL DESCRIPTION:
438 0495 1 Convert FORTRAN error code number to 32-bit VAX-11 error code.
439 0496 1 The following SIGNAL_STOP arguments are obtained from the
440 0497 1 argument list only since no LUB/ISB/RAB yet:
441 0498 1
442 0499 1 VAX-11 error code:
443 0500 1 ST$$V_SEVERITY = ST$$K_SEVERE
444 0501 1 ST$$V_CODE = FORTRAN error number
445 0502 1 ST$$V_FAC_SP = 1 (facility specific error messages
446 0503 1 ST$$V_FAC_NO = FORTRAN facility no. (FOR$$K_FAC_NO)
447 0504 1 3 = No. of following FAO arguments
448 0505 1 FORTRAN unit number if present or zero
449 0506 1 File name string descriptor address or 0 if no FAB
450 0507 1 User PC of call to library (set to 0 here, rewritten by handler before RESIGNAL)
451 0508 1 RMS error code from FAB if present
452 0509 1 System error code from FAB if present
453 0510 1
454 0511 1 FORMAL PARAMETERS:
455 0512 1
456 0513 1 FORT_ERR_NO.rlu.v FORTRAN error code (0:120) or 32-bit cond value
457 0514 1 32-bit VAX-11 error code with LH already set.
458 0515 1 [FORT_LUN.rlu.v] Optional unit number, 0 used if not present
459 0516 1 [FAB.rbu.ra] Address of FAB if present
460 0517 1
461 0518 1 IMPLICIT INPUTS:
462 0519 1
463 0520 1 NONE
464 0521 1
465 0522 1 IMPLICIT OUTPUTS:
466 0523 1
467 0524 1 NONE
468 0525 1
469 0526 1 COMPLETION CODES:
470 0527 1
471 0528 1 NONE
472 0529 1
473 0530 1 SIDE EFFECTS:
474 0531 1
475 0532 1 -- Converts FORTRAN error code to 32-bit VAX-11 error code and SIGNAL_STOPs.
476 0533 1 --
```

```
478 0534 2 BEGIN
479 0535 2 LOCAL
480 0536 2 VAX_11_COND_VAL: BLOCK[4, BYTE], ! 32-bit VAX-11 error code
481 0537 2 NAME_DSC : DSC$DESCRIPTOR, ! File name descriptor
482 0538 2 STS, ! RMS error status
483 0539 2 STV; ! System error status
484 0540 2 MAP
485 0541 2 FORT_ERR_NO: BLOCK[, BYTE], ! MAKE 32-bit VAX-11 error code
486 0542 2 FAB : REF BLOCK [, BYTE]; ! FAB is address of FAB
487 0543 2 BUILTIN
488 0544 2 ACTUALCOUNT; ! Actual no. of parameters
489 0545 2
490 0546 2
491 0547 2 !+
492 0548 2 ! Convert FORTRAN error code to 32-bit VAX-11 error code unless
493 0549 2 ! already converted by the caller. Conversion is done
494 0550 2 ! by copying FORTRAN error number to code field,
495 0551 2 ! setting the severity code to SEVERE,
496 0552 2 ! setting the facility code to FOR$K FAC_NO,
497 0553 2 ! and setting the facility specific bit (ST$V_FAC_SP).
498 0554 2 -
499 0555 2 VAX_11_COND_VAL = COND_VALUE (.FORT_ERR_NO);
500 0556 2
501 0557 2 !+
502 0558 2 ! If FAB argument is present, retrieve RMS and SYSTEM error codes.
503 0559 2 -
504 0560 2
505 0561 2 IF ACTUALCOUNT () GTRU 2
506 0562 2 THEN
507 0563 2 BEGIN
508 0564 2 STS = (IF .FAB [FAB$L_STS] THEN 0 ELSE .FAB [FAB$L_STS]);
509 0565 2 STV = (IF .FAB [FAB$L_STV] THEN 0 ELSE .FAB [FAB$L_STV]);
510 0566 2 END
511 0567 2 ELSE
512 0568 2 BEGIN
513 0569 2 STS = 0;
514 0570 2 STV = 0;
515 0571 2 END;
516 0572 2
517 0573 2 !+
518 0574 2 ! Save FORTRAN error #, RMS STS, RMS STV, logical unit number, and VAX-11 condition value.
519 0575 2 ! If FORT_LUN not present, use 0 (e.g., INVALID ARG TO FORTRAN I/O SYSTEM)
520 0576 2 -
521 0577 2
522 0578 2 FOR$ERRSNS SAV (.FORT_ERR_NO, .STS, .STV,
523 0579 2 (IF ACTUALCOUNT () GTRU 1 THEN .FORT_LUN ELSE 0), .VAX_11_COND_VAL);
524 0580 2
525 0581 2 !+
526 0582 2 ! Set up file name descriptor
527 0583 2 -
528 0584 2 NAME_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
529 0585 2 NAME_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
530 0586 2 IF ACTUALCOUNT () GTRU 2
531 0587 2 THEN
532 0588 2 BEGIN
533 0589 2 IF .FAB [FAB$L_NAM] NEQ 0
534 0590 2 THEN
```

```

535      0591  4      BEGIN
536      0592  4      LOCAL
537      0593  4      NAM : REF BLOCK [BYTE];          ! NAM block
538      0594  4      NAM = .FAB [FAB$L_NAM];
539      0595  4      IF .NAM [NAM$B_RSC] NEQ 0
540      0596  4      THEN
541      0597  5      BEGIN
542      0598  5      NAME_DSC [DSC$W_LENGTH] = .NAM [NAM$B_RSL];
543      0599  5      NAME_DSC [DSC$A_POINTER] = .NAM [NAM$[RSA]];
544      0600  5      END
545      0601  4      ELSE IF .NAM [NAM$B_ESL] NEQ 0
546      0602  4      THEN
547      0603  5      BEGIN
548      0604  5      NAME_DSC [DSC$W_LENGTH] = .NAM [NAM$B_ESL];
549      0605  5      NAME_DSC [DSC$A_POINTER] = .NAM [NAM$[ESA]];
550      0606  5      END
551      0607  4      ELSE
552      0608  5      BEGIN
553      0609  5      NAME_DSC [DSC$W_LENGTH] = .FAB [FAB$B_FNS];
554      0610  5      NAME_DSC [DSC$A_POINTER] = .FAB [FAB$[FNA]];
555      0611  4      END;
556      0612  4      END
557      0613  3      ELSE
558      0614  4      BEGIN
559      0615  4      NAME_DSC [DSC$W_LENGTH] = .FAB [FAB$B_FNS];
560      0616  4      NAME_DSC [DSC$A_POINTER] = .FAB [FAB$[FNA]];
561      0617  3      END;
562      0618  3      END
563      0619  2      ELSE
564      0620  3      BEGIN
565      0621  3      NAME_DSC [DSC$W_LENGTH] = 0;
566      0622  3      NAME_DSC [DSC$A_POINTER] = 0;
567      0623  2      END;
568      0624  2
569      0625  2      !+
570      0626  2      ! Call LIB$STOP to SIGNAL_STOP the error
571      0627  2      ! Order of args is same as defined in FPAR.MDL for use with SYSS$PUT_MESSAGE
572      0628  2      !-
573      0629  2
574      0630  2      LIB$STOP (
575      0631  2      .VAX_11_COND_VAL,          ! 32-bit VAX-11 error code
576      0632  2      K_NO_FAB_SIGARG,         ! no. of FAB arguments following in FORTRAN error message
577      0633  2      .FORT_LUN,              ! FORTRAN logical unit number
578      0634  2      NAME_DSC,                ! file name descriptor
579      0635  2      0,                       ! Leave room for user PC to be filled in
580      0636  2      ! by FORTRAN specific handler established on user call
581      0637  2      .STS,                    ! RMS error code
582      0638  2      .STV);                  ! SYSTEM error code
583      0639  2
584      0640  2      !+
585      0641  2      ! Return
586      0642  2      !-
587      0643  2
588      0644  2      RETURN
589      0645  1      END;                    ! End of FOR$$SIG_NO_LUB routine

```

			001C	00000	.ENTRY	FOR\$\$SIG_NO_LUB, Save R2,R3,R4	:	0486
	5E		08	C2 00002	SUBL2	#8, SP	:	
		04	AC	DD 00005	PUSHL	FORT_ERR_NO	:	0555
0000V	CF		01	FB 00008	CALLS	#1, COND_VALUE	:	
	54		50	D0 0000D	MOVL	R0, VAX_T1_COND_VAL	:	
	02		6C	91 00010	CMPB	(AP), #2	:	0561
			1A	1B 00013	BLEQU	3\$	:	
	50	0C	AC	D0 00015	MOVL	FAB, R0	:	0564
	04	08	A0	E9 00019	BLBC	8(R0), 1\$	:	
			53	D4 0001D	CLRL	STS	:	
			04	11 0001F	BRB	2\$	:	
	53	08	A0	D0 00021	MOVL	8(R0), STS	:	
	08	0C	A0	E8 00025	BLBS	12(R0), 4\$	:	0565
	52	0C	A0	D0 00029	MOVL	12(R0), STV	:	
			04	11 0002D	BRB	5\$	:	0561
			53	D4 0002F	CLRL	STS	:	0569
			52	D4 00031	CLRL	STV	:	0570
			54	DD 00033	PUSHL	VAX_11_COND_VAL	:	0579
	01		6C	91 00035	CMPB	(AP), #1	:	
			05	1B 00038	BLEQU	6\$	:	
		08	AC	DD 0003A	PUSHL	FORT_LUN	:	
			02	11 0003D	BRB	7\$	:	
			7E	D4 0003F	CLRL	-(SP)	:	
			52	DD 00041	PUSHL	STV	:	0578
			53	DD 00043	PUSHL	STS	:	
		04	AC	DD 00045	PUSHL	FORT_ERR_NO	:	
00000000G	00		05	FB 00048	CALLS	#5, FOR\$\$ERRSNS_SAV	:	
	02	010E	8F	B0 0004F	MOVW	#270, NAME_DSC+2	:	0585
			6C	91 00055	CMPB	(AP), #2	:	0586
			38	1B 00058	BLEQU	10\$	:	
	51	0C	AC	D0 0005A	MOVL	FAB, R1	:	0589
		28	A1	D5 0005E	TSTL	40(R1)	:	
			24	13 00061	BEQL	9\$	:	
	50	28	A1	D0 00063	MOVL	40(R1), NAM	:	0594
		03	A0	95 00067	TSTB	3(NAM)	:	0595
			0B	13 0006A	BEQL	8\$	:	
	6E	03	A0	9B 0006C	MOVZBW	3(NAM), NAME_DSC	:	0598
04	AE	04	A0	D0 00070	MOVL	4(NAM), NAME_DSC+4	:	0599
			20	11 00075	BRB	11\$	:	0595
		0B	A0	95 00077	TSTB	11(NAM)	:	0601
			0B	13 0007A	BEQL	9\$	:	
	6E	0B	A0	9B 0007C	MOVZBW	11(NAM), NAME_DSC	:	0604
04	AE	0C	A0	D0 00080	MOVL	12(NAM), NAME_DSC+4	:	0605
			10	11 00085	BRB	11\$	:	0601
	6E	34	A1	9B 00087	MOVZBW	52(R1), NAME_DSC	:	0615
04	AE	2C	A1	D0 0008B	MOVL	44(R1), NAME_DSC+4	:	0616
			05	11 00090	BRB	11\$	:	0586
			6E	B4 00092	CLRW	NAME_DSC	:	0621
		04	AE	D4 00094	CLRL	NAME_DSC+4	:	0622
			52	DD 00097	PUSHL	STV	:	0638
			53	DD 00099	PUSHL	STS	:	0637
			7E	D4 0009B	CLRL	-(SP)	:	0630
		0C	AE	9F 0009D	PUSHAB	NAME_DSC	:	
		08	AC	DD 000A0	PUSHL	FORT_LUN	:	0633

FOR\$\$SIGNAL  
1-007

FORTTRAN SIGNAL, SIGNAL\_STOP and SIG\_NO\_LUB

L 1  
16-Sep-1984 00:44:51  
14-Sep-1984 12:32:44

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

Page 20  
(10)

FO  
1-

03 DD 000A3  
54 DD 000A5  
07 FB 000A7  
04 000AE

PUSHL #3  
PUSHL VAX\_11\_COND\_VAL  
CALLS #7,-LIB\$STOP  
RET

: 0630  
: 0631  
:  
: 0645

; Routine Size: 175 bytes, Routine Base: \_FOR\$CODE + 0113

```

591 0646 1 ROUTINE COND_VALUE ( ! Internal routine to convert from FORTRAN error #
592 0647 1           ! to VAX-11 condition value
593 0648 1     FORT_ERR_NO)      ! Value of FORTRAN error # (0:120) or 32-bit cond value
594 0649 1     =                ! Value is 32-bit VAX-11 condition value
595 0650 1
596 0651 1  +-+
597 0652 1  FUNCTIONAL DESCRIPTION:
598 0653 1      Converts from FORTRAN error number to 32-bit VAX-11 condition value
599 0654 1      complete with proper severity and all other fields set.
600 0655 1      If already a 32-bit condition value (ie GTRU FORSK ERR_MAX),
601 0656 1      no conversions is done. Instead the FORTRAN error # is FORSK_NOTFORSPE
602 0657 1      which has a value of 1 and indicates a non-FORTRAN specific error.
603 0658 1
604 0659 1  FORMAL PARAMETERS:
605 0660 1
606 0661 1     FORT_ERR_NO      ! Value of FORTRAN error # (0:120) or 32-bit cond value
607 0662 1
608 0663 1  IMPLICIT INPUTS:
609 0664 1
610 0665 1     NONE
611 0666 1
612 0667 1  IMPLICIT OUTPUTS:
613 0668 1
614 0669 1     NONE
615 0670 1
616 0671 1  ROUTINE VALUE:
617 0672 1  COMPLETION CODES:
618 0673 1
619 0674 1     32-bit VAX-11 condition value.
620 0675 1
621 0676 1  SIDE EFFECTS:
622 0677 1
623 0678 1     NONE
624 0679 1  --
625 0680 1
626 0681 2  BEGIN
627 0682 2  MAP
628 0683 2     FORT_ERR_NO: BLOCK [4, BYTE];      ! Could be a condition value
629 0684 2  LOCAL
630 0685 2     VAX_11_COND_VAL: BLOCK [4, BYTE];  ! 32-bit VAX-11 error condition value
631 0686 2
632 0687 2  +-+
633 0688 2  Convert FORTRAN error code to 32-bit VAX-11 error code, unless already
634 0689 2  a 32-bit condition value (some other facility than FOR$ in LH).
635 0690 2  Conversion is done by copying FORTRAN error number to code field,
636 0691 2  setting the severity code to SEVERE, except error 63 (OUTPUT CONVERIOSN ERROR)
637 0692 2  in which case the severity is set to ERROR.
638 0693 2  Thus the user must do something explicit in order to continue
639 0694 2  for all errors, except 63 (but it has ***s so error flagged).
640 0695 2  Therefore the user will not inadverantly use data which had errors in it.
641 0696 2  setting the facility code to FORSK FAC NO,
642 0697 2  and setting the facility specific bit (STISSV_FAC_SP).
643 0698 2  -
644 0699 2
645 0700 2  IF .FORT_ERR_NO LEQU FORSK_MAX_ERR
646 0701 2  THEN
647 0702 2     BEGIN

```

```

: 648      0703 3      VAX_11_COND_VAL = 0;
: 649      0704 4      VAX_11_COND_VAL[STSSV_SEVERITY] = (IF .FORT_ERR_NO EQL FOR$K_OUTCONERR
: 650      0705 4      THEN
: 651      0706 4      ST$K_ERROR
: 652      0707 4      ELSE
: 653      0708 3      ST$K_SEVERE);
: 654      0709 3      VAX_11_COND_VAL[STSSV_CODE] = .FORT_ERR_NO;
: 655      0710 3      VAX_11_COND_VAL[STSSV_FAC_SP] = 1;
: 656      0711 3      VAX_11_COND_VAL[STSSV_FAC_NO] = FOR$K_FAC_NO;
: 657      0712 3      END
: 658      0713 2      ELSE
: 659      0714 2      VAX_11_COND_VAL = .FORT_ERR_NO;
: 660      0715 2
: 661      0716 2      RETURN .VAX_11_COND_VAL
: 662      0717 1      END;

```

! End of COND\_VALUE routine

```

                                0000 0000 COND_VALUE:
                                .WORD Save nothing
0000005D 8F 04 AC D1 00002      CMPL FORT_ERR_NO, #93
                                BGTRU 3$
                                51 D4 0000A      CLRL VAX_11_COND_VAL
                                3F 04 AC D1 0000E      CMPL FORT_ERR_NO, #63
                                05 12 00012      BNEQ 1$
                                50 02 D0 00014      MOVL #2, R0
                                03 11 00017      BRB 2$
                                50 04 D0 00019 1$:      MOVL #4, R0
                                51 00 50 F0 0001C 2$:      INSV R0, #0, #3, VAX_11_COND_VAL
                                51 0C 03 04 AC F0 00021      INSV FORT_ERR_NO, #3, #T2, VAX_11_COND_VAL
                                51 8000 8F A8 00027      BISW2 #32768, VAX_11_COND_VAL
                                51 0C 10 18 F0 0002C      INSV #24, #16, #T2, VAX_T1_COND_VAL
                                04 11 00031      BRB 4$
                                51 04 AC D0 00033 3$:      MOVL FORT_ERR_NO, VAX_11_COND_VAL
                                50 51 D0 00037 4$:      MOVL VAX_T1_COND_VAL, R0
                                04 0003A      RET

```

; Routine Size: 59 bytes, Routine Base: \_FOR\$CODE + 01C2

```

: 663      0718 1 END
: 664      0719 0 ELUDOM

```

! End of module

PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$CODE	509	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)



Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255SDUA28:[SYSLIB]STARLET.L32;1	9776	27	0	581	00:01.0
-\$255SDUA28:[FORRTL.OBJ]FORLIB.L32;1	711	186	26	52	00:00.6
-\$255SDUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

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

: Size: 509 code + 0 data bytes  
: Run Time: 00:15.6  
: Elapsed Time: 00:39.4  
: Lines/CPU Min: 2772  
: Lexemes/CPU-Min: 16368  
: Memory Used: 144 pages  
: Compilation Complete



