



```

CCCCCCCC 000000 BBBB8888 DDDDDDDD BBBB8888 EEEEEEEEEE XX XX CCCCCCCC EEEEEEEEEE
CCCCCCCC 000000 BBBB8888 DDDDDDDD BBBB8888 EEEEEEEEEE XX XX CCCCCCCC EEEEEEEEEE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BBBB8888 DD DD BBBB8888 EEEEEEEE XX XX CC CC EEEEEEEE
CC 00 00 BBBB8888 DD DD BBBB8888 EEEEEEEE XX XX CC CC EEEEEEEE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CC 00 00 BB BB DD DD BB BB EE XX XX CC CC EE
CCCCCCCC 000000 BBBB8888 DDDDDDDD BBBB8888 EEEEEEEEEE XX XX CCCCCCCC EEEEEEEEEE
CCCCCCCC 000000 BBBB8888 DDDDDDDD BBBB8888 EEEEEEEEEE XX XX CCCCCCCC EEEEEEEEEE

```

```

LL 111111 SSSSSSSS
LL 111111 SSSSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SSSSSS
LL 11 SSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LLLLLLLLLLLL 111111 SSSSSSSS
LLLLLLLLLLLL 111111 SSSSSSSS

```

```

1 0001 0 MODULE COB$DBEXCEPTION(
2 0002 0 IDENT = '1-011' ! file:COBDBEXCE.B32 Edit:STAN1011
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 * ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 * TRANSFERRED. *
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 * CORPORATION. *
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1 **
30 0030 1 FACILITY: COBOL SUPPORT
31 0031 1
32 0032 1 ABSTRACT
33 0033 1
34 0034 1 This procedure is called from compiled code when a data base
35 0035 1 exception condition occurs. This procedure looks for an applicable
36 0036 1 USE procedure to handle the data base exception condition.
37 0037 1 If one is not found, it invokes LIB$STOP to handle the data base
38 0038 1 exception condition.
39 0039 1
40 0040 1
41 0041 1 ENVIRONMENT: Vax-11 User Mode
42 0042 1
43 0043 1 AUTHOR: RKR , CREATION DATE: 11-FEB-1981
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1 1-001 - Original Skeleton. RKR 11-FEB-1981
48 0048 1 1-002 - Actual code added. LB 11-MAR-1981
49 0049 1 1-003 - Took out code that checked the DB code field (COB$B_USE CODE
50 0050 1 equal to COB$K_DBUSE CODE) This code was moved to
51 0051 1 COB$$HANDLER. -LB 16-MAR-81
52 0052 1 1-004 - Added diagrams of the data structures involved in COB$DBEXCEPTION.
53 0053 1 Also compacted code and changed macro definition name to have a
54 0054 1 DBMS prefix. LB 19-MAR-81
55 0055 1 1-005 - Replaced arbitrary signalling value for no DB USE procedure found
56 0056 1 with appropriate symbol name now defined in COBMSGDEF. Added
57 0057 1 corresponding entry in the EXTERNAL LITERAL declarations for

```

```

58 0058 1 |
59 0059 1 | 1-006 - this module. LB 24-MAR-81
60 0060 1 | Changed name of data base external literal to correspond to change
61 0061 1 | made in COBMSG.MDL. Changed return status of SSS_CONTINUE to
62 0062 1 | SSS_NORMAL. Changed calls to SIGNAL_STOP to be to LIB$STOP for
63 0063 1 | consistency reasons. Changed calls to LIB$SIGNAL to now take an
64 0064 1 | FAO parameter (as is syntactically correct) even though the
65 0065 1 | parameter that is getting passed is not an FAO parameter (note that
66 0066 1 | the !+ directive in the message text will ignore it). Added code
67 0067 1 | back into this routine that had previously been taken out (refer
68 0068 1 | to revision history 1-003); that code now resides in both places
69 0069 1 | (here and in COB$HANDLER). Added heaps of comments and optimized
70 0070 1 | code by encasing calls to LIB$STOP in BEGIN-END blocks. LB 16-APR-81
71 0071 1 | 1-007 - Changed search code for a DB USE procedure due to hidden design flaws
72 0072 1 | in the original search algorithm. Code now saves the first match of
73 0073 1 | a USE procedure and continues searching through the USE list for a
74 0074 1 | locally defined one. If one is found, then the local one is
75 0075 1 | invoked; otherwise, the saved procedure is invoked. Note that this
76 0076 1 | code is executed only for the case where there is an ON OTHER clause
77 0077 1 | (meaning that COB$L_USE_LIT = 0) and when the looping count has not
78 0078 1 | reached its limit. LB 21-APR-81
79 0079 1 | 1-008 - Changed check of looping count to check .I instead of
80 0080 1 | .DBUSE[COB$B_DBUSE CNT]. Moved code that returned SSS_NORMAL to
81 0081 1 | follow directly after the call to LIB$SIGNAL instead of after the
82 0082 1 | END directive. Added code to return SSS_NORMAL within the
83 0083 1 | BEGIN-END blocks at every occurrence of an invocation of a USE
84 0084 1 | procedure to correct flow problems. LB 11-MAY-81.
85 0085 1 | 1-009 - Minor improvements to generated code. PDG 9-AUG-81
86 0086 1 | 1-010 - Declare LIB$SIGNAL external. SBL 2-Dec-1981
87 0087 1 | 1-011 - Remove informational errors. STAN 24-Jul-1984.
88 0088 1 | --
89 0089 1 |
90 0090 1 | !<BLF/PAGE>

```

```

92      0091 1 !+
93      0092 1 ! SWITCHES:
94      0093 1 !-
95      0094 1
96      0095 1 SWITCHES ADDRESSING MODE
97      0096 1 (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
98      0097 1
99      0098 1 !+
100     0099 1 ! LINKAGES:
101     0100 1
102     0101 1 ! NONE
103     0102 1 !-
104     0103 1
105     0104 1 !+
106     0105 1 ! TABLE OF CONTENTS:
107     0106 1 !-
108     0107 1
109     0108 1 FORWARD ROUTINE
110     0109 1 COB$DBEXCEPTION ;
111     0110 1
112     0111 1 !+
113     0112 1 ! INCLUDE FILES:
114     0113 1
115     0114 1 ! Note that there is no require file for the data base
116     0115 1 ! exception codes. These will be provided during link
117     0116 1 ! time as link-time constants.
118     0117 1 !-
119     0118 1
120     0119 1
121     0120 1 REQUIRE 'RTLIN:COBDEF'; ! To find USE procedures
122     0562 1 REQUIRE 'RTLIN:RTLPSECT' ; ! Macros for defining psects
123     0657 1 LIBRARY 'RTLSTARLE';
124     0658 1
125     0659 1 !+
126     0660 1 ! MACROS
127     0661 1 !-
128     0662 1
129     0663 1 MACRO
130     0664 1
131     0665 1 DBMS_COND_VAL = 4,0,32,0%; ! Data Base Condition Value (status code)

```

```
133 0666 1 !+
134 0667 1 EQUATED SYMBOLS
135 0668 1
136 0669 1 NONE
137 0670 1 -
138 0671 1
139 0672 1 !+
140 0673 1
141 0674 1 PSECT DECLARATIONS:
142 0675 1 -
143 0676 1
144 0677 1 DECLARE_PSECTS (COB) ; ! Declare psects for COB$ facility
145 0678 1
146 0679 1 !+
147 0680 1 EXTERNAL REFERENCES:
148 0681 1 -
149 0682 1
150 0683 1 EXTERNAL ROUTINE
151 0684 1 LIB$SIGNAL, ! Signal
152 0685 1 LIB$STOP, ! Signal_stop
153 0686 1 COB$$INVOKE_USE : NOVALUE; ! Invoke_USE procedure
154 0687 1
155 0688 1
156 0689 1 EXTERNAL LITERAL
157 0690 1
158 0691 1 !+
159 0692 1 Condition codes we need
160 0693 1
161 0694 1 Note that the !+ directive for the data base literal indicates to ignore
162 0695 1 the next parameter in the LIB$SIGNAL parameter list. This is needed in
163 0696 1 the case where the call to LIB$SIGNAL passes a condition value and an
164 0697 1 address of a USE procedure which gets checked in COB$HANDLER. The address
165 0698 1 of the USE procedure is not a FAO parameter, but an FAO count must still
166 0699 1 be included (as is syntactically correct) in the call to LIB$SIGNAL in
167 0700 1 the case that COB$HANDLER resorts to re-signalling the error, causing an
168 0701 1 error message to be printed at user level. At that point, the address of
169 0702 1 the USE procedure parameter must not be treated as an FAO parameter -
170 0703 1 therefore the !+ takes care of it (by ignoring the parameter).
171 0704 1
172 0705 1 -
173 0706 1
174 0707 1 COB$_LSTHNDLDB, ! Lost handler for data base exception - environment corrupted !+
175 0708 1 OT$$_FATINTERR; ! Fatal Internal Error
```

```

177 0709 1 GLOBAL ROUTINE COB$DBEXCEPTION (
178 0710 1
179 0711 1     DBMS_STAT           !\Ptr to a block which contains
180 0712 1                                     !/the condition value & error msg.
181 0713 1                                     ) =
182 0714 1
183 0715 1 !++
184 0716 1 ! FUNCTIONAL DESCRIPTION:
185 0717 1
186 0718 1     This procedure is called from compiled code when a data base
187 0719 1     exception occurs when accessing the data base. This procedure
188 0720 1     looks for an applicable USE procedure to handle the error; if
189 0721 1     it can't find one, it LIB$STOP's. If it finds one defined in
190 0722 1     the local program, it invokes the USE procedure; if it is not
191 0723 1     defined in the local program, it signals the data base exception
192 0724 1     condition which should then get processed in COB$HANDLER.
193 0725 1
194 0726 1
195 0727 1 ! CALLING SEQUENCE:
196 0728 1
197 0729 1     COB$DBEXCEPTION (dbms_stat.rr.r)
198 0730 1
199 0731 1 ! FORMAL PARAMETERS:
200 0732 1
201 0733 1     DBMS_STAT.rr.r       Ptr to a block which contains the error
202 0734 1                                     and the condition value to be LIB$SIGNAL'ed
203 0735 1                                     (i.e. a message vector)
204 0736 1
205 0737 1 ! IMPLICIT INPUTS:
206 0738 1
207 0739 1     NONE
208 0740 1
209 0741 1 ! IMPLICIT OUTPUTS:
210 0742 1
211 0743 1     An error message may be signalled.
212 0744 1
213 0745 1 ! COMPLETION CODE:
214 0746 1
215 0747 1     Returns SSS_NORMAL if success (LSB = 1)
216 0748 1     Otherwise, returns a zero.
217 0749 1
218 0750 1 ! SIDE EFFECTS:
219 0751 1
220 0752 1     Calls LIB$STOP if input parameters are insufficient or invalid.
221 0753 1     Also invokes LIB$STOP if the frame pointer equals 0 (indicating
222 0754 1     serious problems).
223 0755 1
224 0756 1 ! NOTES:
225 0757 1
226 0758 1     For more information on the message vector, refer to
227 0759 1     the $PUTMSG system service documentation.
228 0760 1
229 0761 1 !--

```

```

: 231 0762 1 | +
: 232 0763 1 |
: 233 0764 1 |
: 234 0765 1 |
: 235 0766 1 |
: 236 0767 1 |
: 237 0768 1 |
: 238 0769 1 |
: 239 0770 1 |
: 240 0771 1 |
: 241 0772 1 |
: 242 0773 1 |
: 243 0774 1 |
: 244 0775 1 |
: 245 0776 1 |
: 246 0777 1 |
: 247 0778 1 |
: 248 0779 1 |
: 249 0780 1 | -
: 250 0781 1 |
: 251 0782 1 | +
: 252 0783 1 |
: 253 0784 1 |
: 254 0785 1 |
: 255 0786 1 |
: 256 0787 1 |
: 257 0788 1 |
: 258 0789 1 |
: 259 0790 1 |
: 260 0791 1 |
: 261 0792 1 |
: 262 0793 1 |
: 263 0794 1 |
: 264 0795 1 |
: 265 0796 1 |
: 266 0797 1 |
: 267 0798 1 |
: 268 0799 1 |
: 269 0800 1 |
: 270 0801 1 |
: 271 0802 1 |
: 272 0803 1 | -

```

Note that the DB USE list is structured as follows:

```

*****
*                               :COB$B_USE_CODE *
*-----*
*           COB$A_DBUSE_PNC           *
*-----*
*           :COB$B_GDBUSE_CNT:COB$B_DBUSE_CNT*
*-----*
*           COB$A_USE_PROC           * BASE OF 1ST DATA BASE ENTRY
*-----*
*           COB$A_USE_EOPR           *
*-----*
*           COB$L_USE_LIT             *
*****

```

Note that the above fields are defined as follows:

```

COB$B_USE_CODE - generic code indicating that the USE list
                pertains to data base exceptions.
COB$A_DBUSE_PNC -address of Perform Nest Counter for declaring
                program.
COB$B_DBUSE_CNT -number of data base USE procedures defined in
                this program. This includes both local and
                global procedures defined in both this program
                and containing programs.
COB$B_GDBUSE_CNT-number of global data base USE procedures
                defined in the local program.
COB$A_USE_PROC - address of data base USE procedure.
COB$A_USE_EOPR - pointer to the end of the Perform Range Block
                for the USE procedure if the entry was defined
                in this program or 0 if it was defined in a
                containing program.
COB$L_USE_LIT  - a data base exception literal or 0 for 'ON OTHER'
                or no 'ON'.

```

274 0804 1  
275 0805 1  
276 0806 1  
277 0807 1  
278 0808 1  
279 0809 1  
280 0810 1  
281 0811 1  
282 0812 1  
283 0813 1  
284 0814 1  
285 0815 1  
286 0816 1  
287 0817 1  
288 0818 1  
289 0819 1  
290 0820 1  
291 0821 1  
292 0822 1  
293 0823 1  
294 0824 1  
295 0825 1  
296 0826 1  
297 0827 1  
298 0828 1  
299 0829 1  
300 0830 1  
301 0831 1  
302 0832 1  
303 0833 1  
304 0834 1  
305 0835 1

Note that this is the DBMS\_STAT block (referred to as a message vector) that is the input parameter to COB\$DBEXCEPTION. Note that this particular example is for one that contains FAO parameters. There can be other formats, where the condition code is either an RMS status or a system service status, which in both cases, do not take any FAO parameters. Also note that the numbers in parentheses are only meaningful for this sample block. The # of longwords in a DBMS\_STAT block can be up to 255. Also, the # of FAO arguments that can be passed in this block can be up to 16 for DBMS.

```
*****  
*          # OF LONGWORDS IN BLOCK (6)          *  
*-----*  
*          DBMS CONDITION CODE (STATUS)          *  
*-----*  
*          0          ; # OF FAO ARGS (4)          *  
*-----*  
*          FAO ARG #1          *  
*-----*  
*          FAO ARG #2          *  
*-----*  
*          FAO ARG #3          *  
*-----*  
*          FAO ARG #4          *  
*****
```

```
307 0836 2 BEGIN
308 0837 2
309 0838 2 BUILTIN
310 0839 2 CALLG,
311 0840 2 ACTUALPARAMETER,
312 0841 2 ACTUALCOUNT,
313 0842 2 FP;
314 0843 2
315 0844 2 MAP
316 0845 2 DBMS_STAT REF BLOCK[,BYTE],
317 0846 2 FP: REF BLOCK[,BYTE];
318 0847 2
319 0848 2 LOCAL
320 0849 2 SFP: REF BLOCK[,BYTE], : Saved FP
321 0850 2 DBUSE: REF BLOCK[,BYTE], : Pointer to DB USE list
322 0851 2 DBUSE ENT: REF BLOCK[,BYTE], : Pointer to DB USE list entry
323 0852 2 SAVE_PNC: VOLATILE, : Saved addr of PNC
324 0853 2 SAVE_EOPR: VOLATILE, : Saved addr of EOPR
325 0854 2 SAVE_SAVED_AP: VOLATILE, : Saved saved AP
326 0855 2 SAVE_ADDR_USELIST: VOLATILE, : Saved addr of USE list
327 0856 2 SAVE_ADDR_USEPROC; : Saved addr of USE procedure
328 0857 2
329 0858 2
330 0859 2 :+
331 0860 2 : Ensure that the DBMS_STAT argument is present
332 0861 2 :-
333 0862 2 IF (IF ACTUALCOUNT() EQL 0 THEN 1 ELSE .DBMS_STAT EQL 0)
334 0863 2 THEN
335 0864 2 BEGIN
336 0865 2 LIB$STOP (OTSS_FATINTERR);
337 0866 2 RETURN 0;
338 0867 2 END;
339 0868 2
340 0869 2
341 0870 2 :+
342 0871 2 : Initialize the local storage for the
343 0872 2 : saved address of the USE procedure
344 0873 2 : for later use.
345 0874 2 :-
346 0875 2 SAVE_ADDR_USEPROC = 0;
347 0876 2
348 0877 2
349 0878 2 :+
350 0879 2 : Search for an appropriate USE procedure.
351 0880 2 :-
352 0881 2 SFP = .FP[SFSL_SAVE_FP];
353 0882 2 IF .SFP EQL 0 : Get saved FP
354 0883 2 THEN : \If FP equals zero,
355 0884 2 BEGIN : then we have
356 0885 2 LIB$STOP (OTSS_FATINTERR); : /serious problems
357 0886 2 RETURN 0;
358 0887 2 END
359 0888 2 ELSE
360 0889 2 BEGIN
361 0890 2 DBUSE = .SFP[COB$A_DB_USE]; : \Fetch address at offset
362 0891 2 : 8 from stack to obtain
363 0892 2 : /a ptr to a DB USE list
```

```

364 0893 3
365 0894 3
366 0895 3
367 0896 4
368 0897 4
369 0898 4
370 0899 4
371 0900 4
372 0901 4
373 0902 4
374 0903 4
375 0904 4
376 0905 4
377 0906 4
378 0907 4
379 0908 5
380 0909 5
381 0910 5
382 0911 5
383 0912 5
384 0913 5
385 0914 5
386 0915 5
387 0916 5
388 0917 5
389 0918 5
390 0919 6
391 0920 6
392 0921 6
393 0922 6
394 0923 7
395 0924 7
396 0925 7
397 0926 7
398 0927 7
399 0928 7
400 0929 7
401 0930 7
402 0931 7
403 0932 7
404 0933 7
405 0934 7
406 0935 8
407 0936 8
408 0937 8
409 0938 9
410 0939 9
411 0940 9
412 0941 9
413 0942 9
414 0943 9
415 0944 9
416 0945 9
417 0946 9
418 0947 9
419 0948 10
-20 0949 10

```

```

IF .DBUSE NEQ 0
THEN
BEGIN

```

```

+ The following check determines if this is a
data base USE list. The COB$B_USE_CODE field
should contain the generic code for the class of
data base exceptions (equal to COB$K_DBUSE_CODE).
-

```

```

IF .DBUSE[COB$B_USE_CODE] EQL COB$K_DBUSE_CODE

```

```

THEN
BEGIN
DBUSE_ENT = DBUSE[COB$A_DBUSE_ENT]; ! Point to 1st DB USE entry

```

```

+ Find an applicable USE procedure. A USE procedure
is applicable if COB$L_USE_LIT equals the data base
exception or if COB$L_USE_LIT equals zero.
-

```

```

DECR I FROM .DBUSE[COB$B_DBUSE_CNT] - 1 TO 0 DO

```

```

BEGIN
IF .DBUSE_ENT[COB$L_USE_LIT] EQL 0 OR
.DBUSE_ENT[COB$L_USE_LIT] EQL .DBMS_STAT[DBMS_COND_VAL]
THEN
BEGIN

```

```

+ If EOPR (ptr to end of perform range block)
not equal to zero, then we know that
the USE procedure is local, and we can invoke
it immediately. Otherwise, the data base
error should be LIB$SIGNAL'ed.
-

```

```

IF .DBUSE_ENT[COB$A_USE_EOPR] NEQ 0

```

```

THEN
BEGIN
IF .DBUSE_ENT[COB$L_USE_LIT] EQL 0
THEN
BEGIN

```

```

+ If the looping count equals zero, (meaning
there aren't any more USE procedures in the
list), then we can invoke this USE procedure.
-

```

```

IF .I EQL 0
THEN
BEGIN
COB$$INVOKE_USE (

```

```

: 421 0950 10
: 422 0951 10
: 423 0952 10
: 424 0953 10
: 425 0954 10
: 426 0955 10
: 427 0956 10
: 428 0957 9
: 429 0958 9
: 430 0959 9
: 431 0960 9
: 432 0961 9
: 433 0962 9
: 434 0963 9
: 435 0964 9
: 436 0965 9
: 437 0966 9
: 438 0967 9
: 439 0968 9
: 440 0969 9
: 441 0970 9
: 442 0971 9
: 443 0972 9
: 444 0973 9
: 445 0974 9
: 446 0975 9
: 447 0976 10
: 448 0977 10
: 449 0978 10
: 450 0979 10
: 451 0980 10
: 452 0981 10
: 453 0982 9
: 454 0983 9
: 455 0984 8
: 456 0985 8
: 457 0986 8
: 458 0987 8
: 459 0988 8
: 460 0989 8
: 461 0990 8
: 462 0991 8
: 463 0992 8
: 464 0993 9
: 465 0994 9
: 466 0995 9
: 467 0996 9
: 468 0997 9
: 469 0998 9
: 470 0999 9
: 471 1000 9
: 472 1001 8
: 473 1002 8
: 474 1003 7
: 475 1004 8
: 476 1005 8
: 477 1006 8

```

```

        .DBUSE_ENT[COB$A_USE_PROC], ! Addr of USE procedure
        .DBUSE, ! Addr of USE list
        .FP[SF$L_SAVE_AP],
        .DBUSE_ENT[COB$A_USE_EOPR], ! EOPR
        .DBUSE[COB$A_DBUSE_PNC]); ! Perform Nest Ctr
RETURN SSS_NORMAL;
END
ELSE
!+
At this point, we have found a USE procedure,
but rather than invoking it immediately, we
save its address and all other context required
to invoke the USE procedure, and keep on
searching for an additional match. This is
due to the fact that the USE list is arranged
in the order of global USE procedures defined
in the local program, followed by local USE
procedures defined in the local program. This
method of saving the context of the 1st matched
USE procedure now ensures that if a local USE
procedure defined in the local program exists
that it will be invoked rather than a globally
defined one.
-
BEGIN
SAVE_ADDR_USEPROC = .DBUSE_ENT[COB$A_USE_PROC];
SAVE_ADDR_USELIST = .DBUSE;
SAVE_SAVED_AP = .FP[SF$L_SAVE_AP];
SAVE_EOPR = .DBUSE_ENT[COB$A_USE_EOPR];
SAVE_PNC = .DBUSE[COB$A_DBUSE_PNC];
END;
END
ELSE
!+
Here we know that there is no 'ON OTHER'
clause (meaning that COB$A_USE_LIT is not
equal to zero), so we can just invoke the
found USE procedure.
-
BEGIN
COB$$INVOKE USE (
        .DBUSE_ENT[COB$A_USE_PROC], ! Addr of USE procedure
        .DBUSE, ! Addr of USE list
        .FP[SF$L_SAVE_AP],
        .DBUSE_ENT[COB$A_USE_EOPR], ! EOPR
        .DBUSE[COB$A_DBUSE_PNC]); ! Perform Nest Ctr
RETURN SSS_NORMAL;
END;
END
ELSE
BEGIN
!+

```

```

478      1007  8
479      1008  8
480      1009  8
481      1010  8
482      1011  8
483      1012  8
484      1013  8
485      1014  8
486      1015  8
487      1016  8
488      1017  8
489      1018  8
490      1019  8
491      1020  8
492      1021  8
493      1022  8
494      1023  8
495      1024  8
496      1025  8
497      1026  8
498      1027  8
499      1028  8
500      1029  8
501      1030  8
502      1031  8
503      1032  8
504      1033  8
505      1034  8
506      1035  8
507      1036  7
508      1037  6
509      1038  6
510      1039  5
511      1040  5
512      1041  5
513      1042  5
514      1043  5
515      1044  5
516      1045  5
517      1046  5
518      1047  5
519      1048  5
520      1049  5
521      1050  5
522      1051  4
523      1052  4
524      1053  4
525      1054  3
526      1055  3
527      1056  3
528      1057  2
529      1058  2
530      1059  2
531      1060  1

```

```

At this point, if "SAVE_ADDR_USEPROC" not
equal to zero, then we had previously found
a USE procedure which is a globally defined
one in the local program, and saved its context.
We can now invoke it.

IF .SAVE_ADDR_USEPROC NEQ 0
THEN
  COB$$INVOKE USE (
    .SAVE_ADDR_USEPROC,      ! Addr of USE procedure
    .SAVE_ADDR_USELIST,     ! Addr of USE list
    .SAVE_SAVED_AP,         ! Saved AP
    .SAVE_EOPR,             ! Addr of EOPR
    .SAVE_PNC)              ! Addr of PNC
ELSE

+
Signal the error here indicating that the USE
procedure is a global one that is in a containing
program. Need to get the FP of the containing
program and check the defined global USE
procedures of the local program there. If found,
then invoke the USE procedure at that level;
otherwise, the error is re-signalled.

LIB$$SIGNAL (COB$_LSTHNDLDB,1,.DBUSE_ENT[COB$_A_USE_PROC]);
RETURN $$NORMAL;
END;
END;
DBUSE_ENT = .DBUSE_ENT + COB$$DBUSE;      ! Step to next entry
END;                                       ! End of DECR loop

+
At this point, we know that there isn't an
applicable USE procedure, so the only recourse
we have is to LIB$STOP. The reason for this is that
the COB$_USE_CODE did not match with the code
provided in COB$_K_DBUSE_CODE. We also could have
arrived here if DBUSE was equal to zero, indicating
there was no DB USE list to begin with.

END;                                       !\End of code block that
                                           ! checks for a COB$_K_DBUSE_CODE
                                           !/match
END;                                       !\End of code dealing with
                                           !/a valid ptr to a DB USE list

CALLG (.DBMS_STAT,LIB$STOP);
END;
RETURN 0
                                           ! Never gets here
                                           !\End of global routine
                                           !/COB$DBEXCEPTION
END;

```

.TITLE COB\$DBEXCEPTION

```

.IDENT \1-011\
.EXTRN LIB$SIGNAL, LIB$STOP
.EXTRN COB$$INVOKÉ USE
.EXTRN COB$_LSTHND[DB, OTSS$_FATINTERR

.PSECT _COB$CODE, NOWRT, SHR, PIC, 2

.ENTRY COB$DBEXCEPTION, Save R2,R3,R4,R5,R6,R7
57 00000000G 00 00FC 00000 MOVAB LIB$STOP, R7 ; 0709
5E          10 02 00009 SUBL2 #16, SP ;
          6C 95 0000C TSTB (AP) ; 0862
          0D 13 0000E BEQL 1$
          04 AC D5 00010 TSTL DBMS_STAT
          08 13 00013 BEQL 1$
          56 D4 00015 CLRL SAVE_ADDR_USEPROC ; 0875
52 1C AE D0 00017 MOVL 28(FP), SFP ; 0881
          0C 12 0001B BNEQ 2$ ; 0882
          00000000G 8F DD 0001D 1$: PUSHL #OTSS$_FATINTERR ; 0885
67          01 FB 00023 CALLS #1, LIB$STOP
          0099 31 00026 BRW 16$ ; 0886
52 F8 A2 D0 00029 2$: MOVL -8(SFP), DBUSE ; 0890
          03 12 0002D BNEQ 4$ ; 0894
          008C 31 0002F 3$: BRW 15$
01          62 91 00032 4$: CMPB (DBUSE), #1 ; 0905
          FB 12 00035 BNEQ 3$
53 0C A2 9E 00037 MOVAB 12(R2), DBUSE_ENT ; 0909
55 08 A2 9A 0003B MOVZBL 8(DBUSE), I ; 0918
          7A 11 0003F BRB 14$
          51 D4 00041 5$: CLRL R1 ; 0920
          08 A3 D5 00043 TSTL 8(DBUSE_ENT)
          04 12 00046 BNEQ 6$
          51 D6 00048 INCL R1
          0B 11 0004A BRB 7$
04 50 04 AC D0 0004C 6$: MOVL DBMS_STAT, R0 ; 0921
04 A0 08 A3 D1 00050 CMPL 8(DBUSE_ENT), 4(R0)
          61 12 00055 BNEQ 13$
54 04 A3 D0 00057 7$: MOVL 4(DBUSE_ENT), R4 ; 0933
          2B 13 0005B BEQL 9$
1A          51 E9 0005D BLBC R1, 8$ ; 0952
          55 D5 00060 TSTL I ; 0946
          16 13 00062 BEQL 8$
56          63 D0 00064 MOVL (DBUSE_ENT), SAVE_ADDR_USEPROC ; 0977
04 6E 52 D0 00067 MOVL DBUSE, SAVE_ADDR_OSELST ; 0978
08 AE 18 AE D0 0006A MOVL 24(FP), SAVE_SAVED_AP ; 0979
OC AE          54 D0 0006F MOVL R4, SAVE_EOPR ; 0980
          04 A2 D0 00073 MOVL 4(DBUSE), SAVE_PNC ; 0981
          3E 11 00078 BRB 13$ ; 0936
          04 A2 DD 0007A 8$: PUSHL 4(DBUSE) ; 0999
          54 DD 0007D PUSHL R4 ; 0998
          20 AE DD 0007F PUSHL 32(FP) ; 0997
          52 DD 00082 PUSHL DBUSE ; 0996
          63 DD 00084 PUSHL (DBUSE_ENT) ; 0995
          12 11 00086 BRB 10$
          56 D5 00088 9$: TSTL SAVE_ADDR_USEPROC ; 1014
          17 13 0008A BEQL 11$
          0C AE DD 0008C PUSHL SAVE_PNC ; 1021

```

		OC	AE	DD	0008F		PUSHL	SAVE_EOPR	:	1020
		OC	AE	DD	00092		PUSHL	SAVE_SAVED_AP	:	1019
		OC	AE	DD	00095		PUSHL	SAVE_ADDR_OSELST	:	1018
			56	DD	00098		PUSHL	SAVE_ADDR_USEPROC	:	1017
00000000G	00		05	FB	0009A	10\$:	CALLS	#5, COB\$\$INVOKE_USE	:	
			11	11	000A1		BRB	12\$	:	1016
			63	DD	000A3	11\$:	PUSHL	(DBUSE_ENT)	:	1034
			01	DD	000A5		PUSHL	#1	:	
00000000G	00	00000000G	8F	DD	000A7		PUSHL	#COB\$ LSTHNDLDB	:	
	50		03	FB	000AD		CALLS	#3, LIB\$SIGNAL	:	
			01	DO	000B4	12\$:	MOVL	#1, R0	:	1035
				04	000B7		RET		:	
	53		0C	CO	000B8	13\$:	ADDL2	#12, DBUSE_ENT	:	1038
	83		55	F4	000BB	14\$:	SOBGEQ	I, 5\$	:	0918
	67	04	BC	FA	000BE	15\$:	CALLG	@DBMS_STAT, LIB\$STOP	:	1056
			50	D4	000C2	16\$:	CLRL	R0	:	1060
				04	000C4		RET		:	

: Routine Size: 197 bytes, Routine Base: \_COB\$CODE + 0000

: 532 1061 1 END  
: 533 1062 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
_COB\$CODE	197	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	3	0	581	00:00.8

COMMAND QUALIFIERS

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

: Size: 197 code + 0 data bytes

COB\$DBEXCEPTION  
1-011

G 2  
16-Sep-1984 00:01:43

VAX-11 Bliss-32 V4.0-742

Page 14

: Run Time: 00:06.7  
: Elapsed Time: 00:30.6  
: Lines/CPU Min: 9581  
: Lexemes/CPU-Min: 25181  
: Memory Used: 115 pages  
: Compilation Complete

0062 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 120 terminal windows, arranged in 10 rows and 12 columns. Each window shows a different screen of a COBOL program. The programs are identified by their names, which are often followed by 'LIS' (likely listing). The programs shown include:

- COBDIVQ LIS
- COBFINDA LIS
- COBDEXCE LIS
- COBEXPI LIS
- COBDEEDIT LIS
- COBDISPLA LIS
- COBESGEN LIS
- COBERROR LIS
- COBDHANDL LIS

The screens within the windows contain various data, including text, numbers, and some graphical elements like bar charts or tables. The overall appearance is that of a multi-processor terminal session from the VAX/VMS era.