

CCCCCCCC	000000	BBBBBBBB	EEEEEEEEEE	XX	XX	PPPPPPPP	IIIIII
CCCCCCCC	000000	BBBBBBBB	EEEEEEEEEE	XX	XX	PPPPPPPP	IIIIII
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CC	00	00	EE	XX	XX	PP	II
CCCCCCCC	000000	BBBBBBBB	EEEEEEEEEE	XX	XX	PP	II
CCCCCCCC	000000	BBBBBBBB	EEEEEEEEEE	XX	XX	PP	II

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	39	HISTORY	; Detailed current edit history
(3)	75	DECLARATIONS	
(4)	212	MULBIG	Packed Multiply of Big numbers
(5)	311	COBSEXPI	Exponentiate intermediate temporary
(9)	611	FINISH	Convert to destination type and return
(10)	779	CONVERT	Internal routine to convert to intermediate

```
0000 1      .TITLE COB$EXPI      COBOL Intermediate Exponentiate
0000 2      .IDENT /1-012/      ; File: COBEXPI.MAR Edit:LGB1012
0000 3      :
0000 4      :*****
0000 5      :
0000 6      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      :*  ALL RIGHTS RESERVED.
0000 9      :*
0000 10     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     :*  TRANSFERRED.
0000 16     :*
0000 17     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     :*  CORPORATION.
0000 20     :*
0000 21     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     :*
0000 24     :*
0000 25     :*****
0000 26     :
0000 27     :
0000 28     :      VERSION: 1
0000 29     :
0000 30     :      HISTORY:
0000 31     :
0000 32     :      AUTHOR:
0000 33     :           Peter D Gilbert, 2-Aug-1979
0000 34     :
0000 35     :      MODIFIED BY:
0000 36     :
0000 37     :
```

```
0000 39      .SBTTL HISTORY      ; Detailed current edit history
0000 40
0000 41 : Edit history for Version 1 of COBINTARI.MAR
0000 42
0000 43 : 1-001 - Original, with input and output multiplexors lifted from COBINTARI.
0000 44 : PDG 2-Aug-1979
0000 45 : 1-002 - Changed handling of negative bases. PDG 22-Sep-1979.
0000 46 : 1-003 - Make routine PIC. RKR 24-SEPT-79
0000 47 : 1-004 - Correct Undefined Exponentiation error code for LIB$SIGNAL
0000 48 : RKR 27-SEPT-79
0000 49 : 1-005 - Remove additional NON-PIC instruction MOVAV 1[R0], R0
0000 50 : RKR 1-OCT-79
0000 51 : 1-006 - Add additional entry point COB$EXPI_OSE, and minor code improvements
0000 52 : PDG 3-OCT-79
0000 53 : 1-007 - Complete implementation of COB$EXPI_OSE. MLJ 09-Oct-79
0000 54 : 1-008 - Replaced OTSS_FATINTERR signals by COB$ INVARG.
0000 55 : Replaced all LIB$SIGNAL calls by LIB$STOP calls.
0000 56 : Cosmetic changes. RKR 21-OCT-79
0000 57 : 1-009 - Added checks for out-of-range CIT's. RKR 30-OCT-79
0000 58 : 1-010 - Fix check for exponent overflow and underflow in FINISH.
0000 59 : Guarantee that a fraction of zero has an exponent of zero.
0000 60 : RKR 23-APR-80
0000 61 : 1-011 - Changed the branch to FINISH in routine COB$EXPI at label 1$:
0000 62 : to a RET instruction since FINISH expects the input argument
0000 63 : to be in the proper format, where in this case, the argument is
0000 64 : in error and therefore it was never put in the format that
0000 65 : FINISH expects it to be. LB 15-APR-81
0000 66 : 1-012 - Avoid undetected overflow of y in the exponentiate loop by adjusting
0000 67 : the scale of y. Also, restore "begin-loop" and "end-loop" comments.
0000 68 : PDG 28-JUL-82
0000 69 : Prevent a negative base with zero exponent from getting the error
0000 70 : COB$_UNDEF_EXP. It should get a result of 1. A negative base
0000 71 : with an exponent less than zero will continue to correctly get the
0000 72 : COB$_UNDEF_EXP error. LGB 17-AUG-82
0000 73 :
```

```
0000 75      .SBTTL DECLARATIONS
0000 76
0000 77      .DSABL  GBL
0000 78
0000 79  :
0000 80  : INCLUDE FILES:
0000 81  :
0000 82
0000 83      $DSCDEF
0000 84      $INTDEF
0000 85
0000 86  :
0000 87  : EXTERNAL SYMBOLS:
0000 88  :
0000 89      .EXTRN COB$CVTWI_R8      ; Word to intermediate
0000 90      .EXTRN COB$CVTLI_R8      ; Longword to intermediate
0000 91      .EXTRN COB$CVTQI_R8      ; Quadword to intermediate
0000 92      .EXTRN COB$CVTFI_R7      ; Floating to intermediate
0000 93      .EXTRN COB$CVTDI_R7      ; Double to intermediate
0000 94      .EXTRN COB$CVTPI_R9      ; Packed to intermediate
0000 95      .EXTRN COB$CVTIW_R8      ; Intermediate to word
0000 96      .EXTRN COB$CVTIL_R8      ; Intermediate to longword
0000 97      .EXTRN COB$CVTIQ_R8      ; Intermediate to quadword
0000 98      .EXTRN COB$CVTIF_R7      ; Intermediate to floating
0000 99      .EXTRN COB$CVTID_R7      ; Intermediate to double
0000 100     .EXTRN COB$CVTIP_R9      ; Intermediate to packed
0000 101     .EXTRN COB$_INVARG      ; Signal -- Invalid arguments
0000 102     .EXTRN COB$_INTRESOPE    ; Intermediate reserved operand
0000 103     .EXTRN COB$_INTEXPUND    ; Intermediate underflow
0000 104     .EXTRN COB$_INTEXPOVE    ; Intermediate overflow
0000 105     .EXTRN COB$_UNDEF_EXP    ; Undefined exponentiation
0000 106     .EXTRN LIB$STOP          ; Signal msg and stop
0000 107
0000 108  :
0000 109  : MACROS:
0000 110  :
0000 111     .MACRO DCL  SYM,SIZ      ; To define offsets of stack temps
0000 112     SYM      =      OFFSET
0000 113     OFFSET   =      OFFSET + SIZ
0000 114     .ENDM
00000000 0000 115     OFFSET =      0
0000 116
0000 117  :
0000 118  : PSECT DECLARATIONS
0000 119  :
00000000 0000 120     .PSECT _COB$CODE      PIC, SHR, LONG, EXE, NOWRT
0000 121
0000 122  :
0000 123  : EQUATED SYMBOLS:
0000 124  :
00000002 0000 125     INT$P_1_FRACT= 2      ; Temporary until Packed supported in MDL
0000 126                                     ; Fraction field offset
0000 127  :+
0000 128  :
0000 129  : OWN STORAGE:
0000 130  :
0000 131  :+
```

```
0000 132
0000 133
0000 134 : log10( 10^k / (10^k-1) ); k=1,2,...,31 ; ln( 1 + 10^(-k) )
0000 135
0000 136 TABLE1 = -16
44 99 40 25 51 67 60 05 49 57 57 04 0000 137 .PACKED +0457574905606751254099441934852 ; +0105360515657826301227500980829
2C 85 34 19 0000
44 97 65 84 00 45 02 54 80 64 43 00 0010 138 .PACKED +0043648054024500846597442222456 ; +0010050335853501441183548857556
6C 45 22 22 001C
65 64 30 91 76 01 74 17 51 34 04 00 0020 139 .PACKED +0004345117740176913064656006946 ; +0001000500333583533500142982252
6C 94 06 60 002C
56 45 38 10 75 80 19 16 43 43 00 00 0030 140 .PACKED +0000434316198075103845560440232 ; +000010000500033358335333500013
2C 23 40 04 003C
52 93 37 01 39 53 66 29 34 04 00 00 0040 141 .PACKED +0000043429665339013793521486461 ; +00000100000500003333583353333
1C 46 86 14 004C
44 75 63 50 90 69 94 42 43 00 00 00 0050 142 .PACKED +0000004342946990506375442129173 ; +000000100000050000033333583333
3C 17 29 21 005C
37 77 79 61 03 45 29 34 04 00 00 00 0060 143 .PACKED +0000000434294503617977370462100 ; +00000001000000050000033333358
0C 10 62 04 006C
42 72 74 40 48 94 42 43 00 00 00 00 0070 144 .PACKED +0000000043429448407472425164385 ; +0000000010000000050000000333333
5C 38 64 51 007C
99 03 12 82 44 29 34 04 00 00 00 00 0080 145 .PACKED +0000000004342944821203990687473 ; +0000000001000000000500000000333
3C 47 87 06 008C
96 24 19 48 94 42 43 00 00 00 00 00 0090 146 .PACKED +0000000000434294481924966551746 ; +0000000000100000000005000000000
6C 74 51 65 009C
54 90 81 44 29 34 04 00 00 00 00 00 00A0 147 .PACKED +0000000000043429448190542330006 ; +0000000000010000000000050000000
6C 00 30 23 00AC
03 19 48 94 42 43 00 00 00 00 00 00 00B0 148 .PACKED +0000000000004342944819034689748 ; +0000000000001000000000000500000
8C 74 89 46 00BC
90 81 44 29 34 04 00 00 00 00 00 00 00C0 149 .PACKED +0000000000000434294481903273542 ; +0000000000000100000000000005000
2C 54 73 32 00CC
19 48 94 42 43 00 00 00 00 00 00 00 00D0 150 .PACKED +00000000000000043429448190325399 ; +00000000000000010000000000000050
9C 39 25 03 00DC
81 44 29 34 04 00 00 00 00 00 00 00 00E0 151 .PACKED +00000000000000004342944819032518 ; +0000000000000000100000000000000
8C 51 32 90 00EC
48 94 42 43 00 00 00 00 00 00 00 00 00F0 152 .PACKED +0000000000000000434294481903251 ; +0000000000000000010000000000000
1C 25 03 19 00FC
44 29 34 04 00 00 00 00 00 00 00 00 0100 153 .PACKED +0000000000000000043429448190325 ; +0000000000000000001000000000000
5C 32 90 81 010C
94 42 43 00 00 00 00 00 00 00 00 00 0110 154 .PACKED +0000000000000000004342944819032 ; +0000000000000000000100000000000
2C 03 19 48 011C
29 34 04 00 00 00 00 00 00 00 00 00 0120 155 .PACKED +0000000000000000000434294481903 ; +0000000000000000000010000000000
3C 90 81 44 012C
42 43 00 00 00 00 00 00 00 00 00 00 0130 156 .PACKED +0000000000000000000043429448190 ; +0000000000000000000001000000000
0C 19 48 94 013C
34 04 00 00 00 00 00 00 00 00 00 00 0140 157 .PACKED +0000000000000000000004342944819 ; +0000000000000000000000100000000
9C 81 44 29 014C
43 00 00 00 00 00 00 00 00 00 00 00 0150 158 .PACKED +0000000000000000000000434294481 ; +0000000000000000000000010000000
1C 48 94 42 015C
04 00 00 00 00 00 00 00 00 00 00 00 0160 159 .PACKED +0000000000000000000000043429448 ; +0000000000000000000000001000000
8C 44 29 34 016C
00 00 00 00 00 00 00 00 00 00 00 00 0170 160 .PACKED +00000000000000000000000004342944 ; +0000000000000000000000000100000
4C 94 42 43 017C
00 00 00 00 00 00 00 00 00 00 00 00 0180 161 .PACKED +00000000000000000000000000434294 ; +0000000000000000000000000010000
4C 29 34 04 018C
00 00 00 00 00 00 00 00 00 00 00 00 0190 162 .PACKED +00000000000000000000000000043429 ; +00000000000000000000000000010000
9C 42 43 00 019C
```


[illegible]

0402	212	.SBTTL	MULBIG	Packed Multiply of Big numbers
0402	213			
0402	214	++		
0402	215			
0402	216	FUNCTIONAL DESCRIPTION:		
0402	217			
0402	218	Perform a multiply of two 31-digit packed numbers, producing a		
0402	219	63-digit 'packed' result.		
0402	220			
0402	221	CALLING SEQUENCE:		
0402	222			
0402	223	MULBIG (MULRADDR.ap, MULDADDR.ap, PRODADDR.ap)		
0402	224			
0402	225	INPUT PARAMETERS:		
0402	226			
0402	227	MULRADDR.ap	Address of Multiplier	
0402	228	MULDADDR.ap	Address of Multiplicand	
0402	229	PRODADDR.ap	Address of Product (32 bytes)	
0402	230			
0402	231	IMPLICIT INPUTS:		
0402	232			
0402	233	The multiplier and multiplicand are modified during the multiply,		
0402	234	but are left unchanged at exit. Overlapping operands will produce		
0402	235	incorrect results.		
0402	236			
0402	237	IMPLICIT OUTPUTS:		
0402	238			
0402	239	NONE		
0402	240			
0402	241	--		
0402	242			
0402	243	.ENTRY	MULBIG, ^M<R2,R3,R4,R5,R8>	; Don't want this to be global
013C	0402	MULBIG: .WORD	^M<R2,R3,R4,R5,R8>	
	0404	245		
00000000	0404	246	A\$ = 0	
00000012	0404	247	B\$ = 18 + A\$	
00000024	0404	248	C\$ = 18 + B\$	
00000030	0404	249	D\$ = 12 + C\$	
0000003C	0404	250	E\$ = 12 + D\$	
	0404	251		
SE 3C C2	0404	252	SUBL2 #E\$,SP	
	0407	253		
58 0C AC D0	0407	254	MOVL 12(AP),R8	
	040B	255		
02 AE 04 BC 1F 34	040B	256	MOVP #31,24(AP), 2+A\$(SP)	
6E 02 AE 05 28	0411	257	MOVC3 #5,2+0+A\$(SP),0+A\$(SP)	
06 AE 07 AE 05 28	0416	258	MOVC3 #5,2+5+A\$(SP),6+A\$(SP)	
50 11 AE F0 8F 8B	041C	259	BICB3 #^XF0,5+12+A\$(SP),R0	
	0422	260	MOVB R0, 5+ 6+A\$(SP)	
	0426	261	MOVB R0, 5+ 0+A\$(SP)	
	042A	262		
14 AE 08 BC 1F 34	042A	263	MOVP #31,28(AP), 2+B\$(SP)	
12 AE 14 AE 05 28	0430	264	MOVC3 #5,2+0+B\$(SP),0+B\$(SP)	
18 AE 19 AE 05 28	0436	265	MOVC3 #5,2+5+B\$(SP),6+B\$(SP)	
50 23 AE F0 8F 8B	043C	266	BICB3 #^XF0,5+12+B\$(SP),R0	
	0442	267	MOVB R0, 5+ 6+B\$(SP)	
	0446	268	MOVB R0, 5+ 0+B\$(SP)	

[illegible]

```

CLRQ      0(R8)
CLRQ      8(R8)
MULP      #11,12+AS(SP),#11,12+BS(SP),#31,16(R8) ; A0xB0

XORB3     5+12+AS(SP), 5+12+BS(SP), R0
BISB3     #^X0C,R0,31(R8)

MULP      #11,12+AS(SP),#11, 6+BS(SP),#23,C$(SP) ; A0xB1

MULP      #11, 6+AS(SP),#11,12+BS(SP),#23,D$(SP) ; A1xB0

ADDP4     #23, DS(SP),#23, C$(SP)
MOVB      11+15(R8), R4
BICB3     #^XF0, 31(R8), 11+15(R8)
ADDP4     #23, C$(SP),#23, 15(R8)
MOVB      R4, 11+15(R8)

MULP      #11,12+AS(SP),#11, 0+BS(SP),#23,C$(SP) ; A0xB2

MULP      #11, 6+AS(SP),#11, 6+BS(SP),#23,D$(SP) ; A1xB1

ADDP4     #23, DS(SP),#23, C$(SP)
MULP      #11, 0+AS(SP),#11,12+BS(SP),#23,D$(SP) ; A2xB0

ADDP4     #23, DS(SP),#23, C$(SP)
MOVB      11+10(R8), R4
BICB3     #^XF0, 31(R8), 11+10(R8)
ADDP4     #23, C$(SP),#23, 10(R8)
MOVB      R4, 11+10(R8)

MULP      #11, 6+AS(SP),#11, 0+BS(SP),#23,C$(SP) ; A1xB2

MULP      #11, 0+AS(SP),#11, 6+BS(SP),#23,D$(SP) ; A2xB1

ADDP4     #23, DS(SP),#23, C$(SP)
MOVB      11+ 5(R8), R4
BICB3     #^XF0, 31(R8), 11+ 5(R8)
ADDP4     #23, C$(SP),#23, 5(R8)
MOVB      R4, 11+ 5(R8)

MULP      #11, 0+AS(SP),#11, 0+BS(SP),#23,C$(SP) ; A2xB2

MOVB      11+ 0(R8), R4
BICB3     #^XF0, 31(R8), 11+ 0(R8)
ADDP4     #23, C$(SP),#23, 0(R8)
MOVB      R4, 11+ 0(R8)

RET

```

```
0525 311 .SBTTL COB$EXPI Exponentiate intermediate temporary
0525 312
0525 313 :++
0525 314 :
0525 315 : FUNCTIONAL DESCRIPTION:
0525 316 :
0525 317 : Accept any two supported data types as input, convert them to
0525 318 : Intermediate, exponentiate them, convert the Intermediate result
0525 319 : to the data type of the output argument, and return.
0525 320 :
0525 321 : 1. If routine is confronted with unknown data type it
0525 322 : SIGNALSTOPS COB$_INVARG.
0525 323 :
0525 324 : 2. If presented with an input CIT which has an overflowed or
0525 325 : underflowed exponent field it SIGNALSTOPS COB$_INTRESOPE.
0525 326 :
0525 327 : 3. If entered at COB$EXPI_OSE (on size error) and
0525 328 : exponentiation can't be done (e.g. exp < 0), returns 0.
0525 329 :
0525 330 : 4. If entered at COB$EXPI and exponentiation can't be done
0525 331 : it SIGNALSTOPS COB$_UNDEXP.
0525 332 :
0525 333 : 5. If exponentiation is performed and
0525 334 : If resulting CIT has overflowed exponent field,
0525 335 : SIGNALSTOP COB$_INTEXPOVE.
0525 336 :
0525 337 : If resulting CIT has underflowed exponent field,
0525 338 : SIGNALSTOP COB$_INTEXPUND.
0525 339 : CALLING SEQUENCE:
0525 340 :
0525 341 : COB$EXPI (BASE.rx.dx, EXPONENT.rx.dx, POWER.wx.dx)
0525 342 : COB$EXPI_OSE (BASE.rx.dx, EXPONENT.rx.dx, POWER.wx.dx)
0525 343 :
0525 344 : INPUT PARAMETERS:
0525 345 :
0525 346 : BASE.rx.dx The operand to the left of the operator
0525 347 : EXPONENT.rx.dx The operand to the right of the operator
0525 348 :
0525 349 : IMPLICIT INPUTS:
0525 350 :
0525 351 : NONE
0525 352 :
0525 353 : OUTPUT PARAMETERS:
0525 354 :
0525 355 : POWER.wx.dx The power BASE ** EXPONENT
0525 356 :
0525 357 : IMPLICIT OUTPUTS:
0525 358 :
0525 359 : Different error handling for different entry points:
0525 360 : COB$EXPI - Call LIB$STOP for bad exponentiation.
0525 361 : COB$EXPI_OSE - Return R0 = 1 or 0 for success or failure.
0525 362 :
0525 363 : FUNCTION VALUE:
0525 364 :
0525 365 : COB$EXPI_OSE - 1 or 0, depending on success or failure.
0525 366 :
0525 367 : SIDE EFFECTS:
```

```
0525 368 :  
0525 369 : NONE  
0525 370 :  
0525 371 : QUOTES:  
0525 372 :  
0525 373 : "The invisible are insane."  
0525 374 : - English translation of a Chinese translation of  
0525 375 : an English proverb.  
0525 376 :  
0525 377 : "The Stone the Builders Rejected".  
0525 378 : - Inscription on Jack London's gravestone.  
0525 379 :--  
0525 380 :  
0525 381 DCL base_it,INT$K_I_LEN ; intermediate temp  
0525 382 DCL exp_it,INT$K_I_LEN ; intermediate temp  
0525 383 DCL res_it,INT$K_I_LEN ; intermediate temp  
0525 384 DCL x, 16 ; for 31 digits  
0525 385 DCL y, 16 ; for 31 digits  
0525 386 DCL z, 32 ; for 63 digits  
0525 387 DCL res_sign,1 ; sign of result  
0525 388 DCL ose, 1 ; remember entry point
```

```

      12 37 0525 390 ERR_0: ; Base = 0
      OE AE 37 0525 391 CMPP4 #INT$K_I_FRACT_D,-
FEC3 CF 01 0527 392 INT$P_I_FRACT+exp_it(SP),-
      06 15 0529 393 #1,P0
      50 01 D0 052D 394 BLEQ ERR_BAD ; If exponent > 0 return 0.0 (= base)
      0235 31 052F 395 MOVL #1,R0
      OD 65 AE E8 0532 396 BRW FINISH
      00000000'8F DD 0535 397 ERR_BAD:
00000000'GF 01 FB 0539 398 BLBS ose(SP),1$ ; Br if to return status
      50 D4 053F 399 PUSHL #COB$_UNDEF_EXP ; Undefined exponentiation
      04 0546 400 CALLS #1,G^CIB$STOP ; Signal it and quit
      04 0548 401 1$: CLRL R0 ; Error
      04 0549 402 RET
      50 01 03FC 0549 403 .ENTRY COB$EXPI_OSE,^M<R2,R3,R4,R5,R6,R7,R8,R9>
      04 11 054B 404 MOVL #1,R0 ; Remember flavor of call
      03FC 054E 405 BRB EXP_J
      0550 406 .ENTRY COB$EXPI,^M<R2,R3,R4,R5,R6,R7,R8,R9>
      0552 407
      0552 408 ; Convert to intermediate
      0552 409
      SE 50 D4 0552 410
      00000066 8F C2 0554 411 EXP_J: CLRL R0
      65 AE 50 90 0554 412 SUBL2 #offset,SP
      50 04 AC D0 055B 413 MOVB R0,ose(SP)
      51 6E 9E 055F 414 MOVL 4(AP),R0
      0336 30 0563 415 MOVAB base_it(SP),R1
      50 08 AC D0 0566 416 BSBW CONVERT
      51 0C AE 9E 0569 417 MOVL 8(AP),R0
      032B 30 056D 418 MOVAB exp_it(SP),R1
      0571 419 BSBW CONVERT
      0574 420
      0574 421 ; Compute the log base 10 of the base
      0574 422
      0574 423
      02 AE 95 0574 424 TSTB base_it+INT$P_I_FRACT(SP) ; See if base is zero
      AC 13 0577 425 BEQL ERR_0 ; What to do? ###
      0579 426
      0579 427 ; Determine the correct sign of the result
      0579 428
      64 AE 0C 90 0579 429 MOVB #^X0C,res_sign(SP) ; Assume positive
      24 AE 0B AE 10 89 057D 430 BISB3 #^X10,<INT$K_I_FRACT_D/2>+INT$P_I_FRACT+base_it(SP),x(SP)
FE66 CF 24 AE 01 35 0583 431 CMPP3 #1,x(SP),P0
      54 0C AE 23 18 058A 432 BGEQ 1314$ ; It is positive
      32 058C 433 CVTWL INT$W_I_EXP+exp_it(SP),R4 ; If exp <= 0
      0590 434
      0590 435 ; BLEQ instruction changed to the BLSS instruction below,
      0590 436 ; see edit 1-012
      0590 437
      24 AE 12 00 12 A3 19 0590 438 BLSS ERR_BAD ; then exponent isn't integral
      54 54 F8 0592 439 ASHP R4,#INT$K_I_FRACT_D,- ; Look at the fractional part
      0E AE 98 12 0595 440 INT$P_I_FRACT+exp_it(SP),#0,#INT$K_I_FRACT_D,x(SP)
      54 12 A2 059B 441 BNEQ ERR_BAD ; If non-zero, then bad
      61 12 54 F8 059D 442 SUBW2 #INT$K_I_FRACT_D,R4 ; Look at the integer part
      04 63 04 E1 05A0 443 ASHP R4,#INT$K_I_FRACT_D,(R1),#0,#1,(R3)
      64 AE 0D 90 05A7 444 BBC #4,(R3),1314$ ; If even, result is positive
      05AB 445 MOVB #^X0D,res_sign(SP) ; Else result is negative
      05AF 446 ;
```

```

      02 AE 12 OC F8 05AF 447 : begin log loop
      24 AE 1E 00 05AF 448 :
      33 AE 0F 88 05B1 449 1314$: ASHP #<30-INT$K_I_FRACT_D>,-
      1D 00 63 1E FF 8F 05B4 450 #INT$K_I_FRACT_D,base_it+INT$P_I_FRACT(SP),-
      1F 00 FE25 CF 01 1E F8 05B8 451 #0,#30,x(SP) ; set x
      44 AE 44 05BC 452 BISB2 #^X0F,15+x(SP) ; make it positive
      56 01 CE 05C3 453 ASHP #-1,#30,(R3),#0,#29,z(SP) ; z <- x/10
      01 24 AE 05C5 454 ASHP #<31-1>,#1,P0,#0,#31,y(SP) ; y <- 0
      FE10 CF 01 25 AE 1D 05CD 455 MNEGL #1,R6 ; k <- 1
      44 AE 25 AE 1D 05D2 456 2$: CMPB x(SP),#1 ; x = 1 ?
      24 AE 1F 44 AE 1D 05D6 457 BGTR 3$ ; br if x > 1
      1D 00 24 AE 1E 56 05D8 458 CMPP4 #29,1+x(SP),#1,P0
      50 1F 56 F9E8 CF40 05E0 459 BEQL 10$ ; br if log calculation is done
      34 AE 1F 56 F9E8 CF40 05E2 460 CMPP3 #29,1+x(SP),z(SP) ; x-z >= 1 ?
      1D 00 24 AE 1E 56 05E8 461 BLSS 4$ ; br if not
      56 01 22 05EA 462 3$: SUBP4 #29,z(SP),#31,x(SP) ; x <- x-z
      34 AE 1F 56 F9E8 CF40 05F1 463 ASHP R6,#30,x(SP),#0,#29,z(SP) ; z <- x / 10^k
      1D 00 24 AE 1E 56 05F8 464 MULL3 #-16,R6,R0 ; k * 16
      34 AE 1F 56 F9E8 CF40 0602 465 ADDP4 #31,tab[e1[R0],#31,y(SP) ; y <- y + log( 10^k/(10^k-1) )
      1D 00 24 AE 1E 56 060B 466 BRB 2$
      56 01 22 060D 467 4$: DECL R6 ; k <- k+1
      34 AE 1F 56 F9E8 CF40 060F 468 ASHP R6,#30,x(SP),#0,#29,z(SP) ; z <- z / 10
      1D 00 24 AE 1E 56 0616 469 BNEQ 2$ ; finished if k gets very large
      34 AE 1F 56 F9E8 CF40 061A 470 10$:
      1D 00 24 AE 1E 56 061A 471 :
      34 AE 1F 56 F9E8 CF40 061A 472 :
      1D 00 24 AE 1E 56 061A 473 :
      34 AE 1F 56 F9E8 CF40 061A 474 :
      1D 00 24 AE 1E 56 061A 475 :
      34 AE 1F 56 F9E8 CF40 061A 476 :
      1D 00 24 AE 1E 56 061A 477 :
      34 AE 1F 56 F9E8 CF40 061A 478 :
      1D 00 24 AE 1E 56 061A 479 E_d = 2
      34 AE 1F 56 F9E8 CF40 061A 480 .IIF GE,INT$K_I_EXP_HI-100, E_d = 4
      1D 00 24 AE 1E 56 061A 481 .IIF LT,INT$K_I_EXP_LO+100, E_d = 4
      34 AE 1F 56 F9E8 CF40 061A 482 .IIF GE,INT$K_I_EXP_HI-10000, E_d = 6
      1D 00 24 AE 1E 56 061A 483 .IIF LT,INT$K_I_EXP_LO+10000, E_d = 6
      34 AE 1F 56 F9E8 CF40 061A 484 :
      1D 00 24 AE 1E 56 061A 485 :
      34 AE 1F 56 F9E8 CF40 061A 486 :
      1D 00 24 AE 1E 56 061A 487 :
      34 AE 1F 56 F9E8 CF40 061A 488 :
      1D 00 24 AE 1E 56 061A 489 :
      34 AE 1F 56 F9E8 CF40 061A 490 begin multiply by exponent
      1F 00 44 AE 03 50 32 0622 491 SUBP4 #31,NINES,#31,y(SP) ; Subtract one (essentially)
      44 AE 03 50 32 0625 492 CVTWL INT$W_I_EXP+base_it(SP),R0 ; need to add this on
      24 AE 0F 8A 062A 493 CVTLP R0,#E_d+1,z(SP)
      1D 00 44 AE 03 50 32 0631 494 ASHP #31-E_d,#E_d+1,z(SP),#0,#31,x(SP)
      24 AE 1F 34 AE 1D 20 0633 495 BICB2 #^X0F,15-<E_d/2>+y(SP) ; Move sign closer in y
      1D 00 44 AE 03 50 32 0637 496 BISB2 #^X0D,15-<E_d/2>+y(SP) ; Remember it's negative now
      24 AE 1F 34 AE 1D 20 063B 497 ADDP4 #31-E_d,y(SP),#31,x(SP) ; x <- y + exponent - 1
      1D 00 44 AE 03 50 32 0642 498
      24 AE 1F 34 AE 1D 20 0642 498
```

COB\$EXPI
1-012

COBOL Intermediate Exponentiate G 15
COB\$EXPI Exponentiate intermediate tempo 15-SEP-1984 23:43:25 VAX/VMS Macro V04-00
6-SEP-1984 10:45:28 [COBRTL.SRC]COBEXPI.MAR;1

Page 13
(6)

OE AE OD
34 AE 1F 00
44 AE 9F
38 AE 9F
2C AE 9F
FDA9 CF 03 FB
0642 499
0644 500
0647 501
064B 502
064E 503
0651 504
0654 505
0659 506
0659 507
0659 508
0659 509
0659 510

ASHP #31-INTSK_I_FRACT_D,-
#INTSK_I_FRACT_D,INTSP_I_FRACT+exp_it(SP),-
#0,#31,y(SP)
PUSHAB z+0(SP)
PUSHAB y+4(SP)
PUSHAB x+8(SP)
CALLS #3,MULBIG
-099.9999 < z < +099.9999
end multiply by exponent
; z <- y * x
; (this is why z needs 63 digs)

```
0659 512 :  
0659 513 :  
0659 514 : We want to grab the integer and fractional parts of this product.  
0659 515 : Unfortunately, the VAX does not (yet) support 63 digit packed numbers.  
0659 516 :  
44 AE 1F 00 3B 0659 517 SKPC #0,#31,z(SP) ; Skip leading zeroes  
54 50 D0 065E 518 movl R0,R4 ; Save # of bytes remaining  
03 50 D1 0661 519 cmpl R0,#3 ; Do we have at least 7 digits?  
16 14 0664 520 bgtr 1001$ ; Branch if so  
50 01 A040 9E 0666 521 movab 1(R0)[R0],R0 ; # of digits in product  
52 07 50 C3 066B 522 subl3 R0,#7,R2 ; Calculate shift amount  
24 AE 07 00 61 50 52 F8 066F 523 ashp R2,R0,(R1),#0,#7,x(SP) ; Shift exponent into x  
19 11 0677 524 brb 1002$ ; Merge with other case  
FEB9 31 0679 525 9191$: BRW ERR_BAD ; Branch point  
03 A1 28 AE 03 A1 90 067C 526 1001$: movb 3(RT),4+x(SP) ; Save these digs a moment  
63 AE FO 8F 8B 0681 527 bicb3 #^XF0,31+z(SP),3(R1) ; Put in the correct sign  
24 AE 61 07 34 0688 528 movp #7,(R1),x(SP) ; The exponent's hidden in here  
03 A1 28 AE 90 068D 529 movb 4+x(SP),3(R1) ; Restore digits  
56 0C AE 32 0692 530 1002$: CVTWL INT$W_I_EXP+exp_it(SP),R6 ; Get the exponent's exponent  
07 00 24 AE 50 BE A644 3E 0696 532 MOVAV -68+E_d(R6)[R4],R0 ; Calculate shift amount (!)  
07 00 24 AE 07 50 F8 069B 533 ASHP R0,#7,x(SP),#0,#7,4+x(SP)  
28 AE D3 1D 06A4 534 BVS 9191$ ; Exponent overflow ####  
51 28 AE 07 36 06A6 535 1004$: CVTPL #7,4+x(SP),R1  
18 AE 51 F7 06AB 536 CVTLW R1,INT$W_I_EXP+res_it(SP)  
C8 1D 06AF 537 BVS 9191$  
06B1 538 :  
06B1 539 : The decimal point is E_d+1+R6 places to the right of z  
06B1 540 :  
54 D4 06B1 541 CLRL R4  
57 03 A6 9E 06B3 542 MOVAB E_d+1(R6),R7  
0D 15 06B7 543 BLEQ 1010$ ; R4 is correct offset from z(SP)  
54 57 FF 8F 78 06B9 544 ASHL #-1,R7,R4  
10 54 D1 06BE 545 CMPL R4,#16  
03 15 06C1 546 BLEQ 1010$ ; R4 is correct offset from z(SP)  
54 10 D0 06C3 547 MOVL #16,R4 ; R4 is correct offset from z(SP)  
06C6 548 1010$: Shift by E_d + 1 + R6 - 2*R4 = R7 - 2*R4  
06C6 549 :  
06C6 550 :  
06C6 551 :  
50 63 AE FO 8F 8B 06C6 552 BICB3 #^XF0,31+z(SP),R0 ; Put in a correct sign  
53 AE44 OF 8A 06CC 553 BICB2 #^XOF,15+z(SP)[R4]  
53 AE44 50 88 06D1 554 BISB2 R0,15+z(SP)[R4]  
50 57 54 FFFFFFFE 8F 7A 06D6 555 EMUL #-2,R4,R7,R0 ; R7 - 2 * R4  
1F 00 44 AE44 1F 50 F8 06DF 556 ASHP R0,#31,z(SP)[R4],#0,#31,x(SP) ; Shift into x  
24 AE 13 18 06E9 557 BGEQ 1011$ ; >= 0 ? D'lovely.  
24 AE 1F FD02 CF 1F 20 06EB 558 ADDP4 #31,NINES,#31,x(SP) ; Add 1.0  
24 AE 1F FCF9 CF 01 20 06F3 559 ADDP4 #1,P1,#31,x(SP)  
18 AE B7 06FB 560 DECW INT$W_I_EXP+res_it(SP) ; Decrease result's exponent  
06FE 561 1011$: 1011$: 561  
06FE 562
```

```
06FE 564 :  
06FE 565 :  
06FE 566 :  
06FE 567 :  
06FE 568 :  
1F 00 FCED CF 01 1D F8 06FE 569 :  
34 AE 0706 :  
54 56 FFFFFFFF 8F D4 0708 570 :  
24 AE FADB CF44 1F C5 070A 571 11$:  
1D 14 0712 572 :  
1F 00 34 AE 1F 56 F8 071A 573 :  
44 AE 071C 574 :  
1E 13 0723 :  
34 AE 1F 44 AE 1F 20 0725 575 :  
24 AE 1F FABC CF44 1F 20 0727 576 :  
D1 11 072E 577 :  
56 D7 0737 578 :  
FCAD CF 01 24 AE 1F D7 0739 579 12$:  
C5 12 073B 580 :  
0743 581 :  
0745 582 20$:  
34 AE F4 8F F8 0745 583 :  
1F 0748 584 :  
01 074B 585 :  
12 074C 586 :  
1A AE 074D 587 :  
06 1C 074F 588 :  
63 01 90 0751 589 :  
18 AE B6 0754 590 :  
0757 591 :  
18 AE B6 0757 592 21$:  
075A 593 :  
075A 594 :  
075A 595 :  
23 AE 0F 8A 075A 596 :  
23 AE 64 AE 88 075E 597 :  
0763 598 :  
0763 599 :  
0763 600 :  
0763 601 :  
5E 18 AE 9E 0763 602 :  
50 01 D0 0767 603 :  
076A 604 :  
076A 605 :  
076A 606 :  
076A 607 :  
076A 608 :  
076A 609 :  
begin exponentiate loop  
+00000 <= x <= +99999  
ASHP #29,#1,P1,#0,#31,y(SP) : y <- 1  
CLRL R6 : k <- 0  
MULL3 #-16,R6,R4 :  
CMPP3 #31,tab[e2[R4],x(SP) : log(1+10^(-k)) > x ?  
BGTR 12$ :  
ASHP R6,#31,y(SP),#0,#31,z(SP) : y <- y + 10^(-k)y  
BEQL 20$ :  
ADDP4 #31,z(SP),#31,y(SP) :  
SUBP4 #31,tab[e2[R4],#31,x(SP) : x <- x - log(1+10^(-k))  
BRB 11$ :  
DECL R6 : k <- k + 1  
CMPP4 #31,x(SP),#1,P0 : if w = 0 then exitloop  
BNEQ 11$ :  
ASHP #INT$K_I_FRACT_D-30,- : Shift into result  
#31,y(SP),- :  
#1,- : (rounded ever so slightly)  
#INT$K_I_FRACT_D,-  
INT$P_I_FRACT+res_it(SP)  
BVC 21$ :  
MOVB #1@<4*<1@INT$K_I_FRACT_D>>,(R3) : Means answer = 1  
INCW INT$W_I_EXP+res_it(SP) : Increment exponent (#1)  
INCW INT$W_I_EXP+res_it(SP) : Increment exponent (#2)  
; Put in the correct sign  
BICB2 #^XOF,INT$K_I_FRACT_D/2+INT$P_I_FRACT+res_it(SP)  
BISB2 res_sign(SP),INT$K_I_FRACT_D/2+INT$P_I_FRACT+res_it(SP)  
end exponentiate loop  
MOVAB res_it(SP),SP : B.O.H.I.  
MOVL #1,R0 : Success. Exaltation.  
BRW FINISH  
All done with the hard part. Now fall through and convert to destination.
```

```
076A 611 .SBTTL FINISH Convert to destination type and return
076A 612
076A 613 :+
076A 614 :
076A 615 : Enter by branch with (SP) containing the intermediate result
076A 616 : and 12(AP) pointing to the descriptor for the destination.
076A 617 :- RO contains routine status
076A 618
076A 619 FINISH:
02 AE 95 076A 620 TSTB INTSP_I_FRACT(SP) ; is fraction zero ?
04 12 076D 621 BNEQ 8$ ; no
6E B4 076F 622 CLRW INTSW_I_EXP(SP) ; force exponent to zero
OE 11 0771 623 BRB 9$ ; bypass overflow and underflow
0773 624 ; checks
0773 625 :+
0773 626 : Check for out-of-range conditions first
0773 627 : We do the check here for all destination type so that we can report
0773 628 : overflow and underflow distinctly. If we allow the flow to go
0773 629 : directly to various COB$CVTI_x routines, what will be reported
0773 630 : is COB$_INTRESOPE (which is not correct -- we just created the
0773 631 : exception and did not access it -- creating an exception should
0773 632 : distinguish between over_ and under_flow)
0773 633 :-
0773 634
0773 635 8$:
0063 8F 6E B1 0773 636 CMPW INTSW_I_EXP(SP), #INTSK_I_EXP_HI ; Bigger than max ?
55 14 0778 637 BGTR 3$ ; Yes, overflow
FF9D 8F 6E B1 077A 638 CMPW INTSW_I_EXP(SP), #INTSK_I_EXP_LO ; Less than min ?
56 19 077F 639 BLSS 5$ ; Yes, underflow
0781 640 9$:
50 DD 0781 641 PUSHL R0 ; Save success status
0783 642 ; Result now at 4(SP)
0783 643
1F 50 0C AC D0 0783 644 MOVL 12(AP),R0
00 02 A0 8F 0787 645 CASEB DSC$B_DTYPE(R0),#0,#31
0202' 078C 646 10$: .WORD BAD_DT-10$ 0 Z
0202' 078E 647 .WORD BAD_DT-10$ 1 V
0202' 0790 648 .WORD BAD_DT-10$ 2 BU
0202' 0792 649 .WORD BAD_DT-10$ 3 WU
0202' 0794 650 .WORD BAD_DT-10$ 4 LU
0202' 0796 651 .WORD BAD_DT-10$ 5 QU
0202' 0798 652 .WORD BAD_DT-10$ 6 B
0058' 079A 653 .WORD 20$-10$ 7 W
0079' 079C 654 .WORD 30$-10$ 8 L
009A' 079E 655 .WORD 40$-10$ 9 Q
00BB' 07A0 656 .WORD 50$-10$ 10 F
00CD' 07A2 657 .WORD 60$-10$ 11 D
0202' 07A4 658 .WORD BAD_DT-10$ 12 FC
0202' 07A6 659 .WORD BAD_DT-10$ 13 DC
0202' 07A8 660 .WORD BAD_DT-10$ 14 T
0202' 07AA 661 .WORD BAD_DT-10$ 15 NU
0202' 07AC 662 .WORD BAD_DT-10$ 16 NL
0202' 07AE 663 .WORD BAD_DT-10$ 17 NLO
0202' 07B0 664 .WORD BAD_DT-10$ 18 NR
0202' 07B2 665 .WORD BAD_DT-10$ 19 NRO
0202' 07B4 666 .WORD BAD_DT-10$ 20 NZ
00DF' 07B6 667 .WORD 70$-10$ 21 P
```

```
0202' 07B8 668 .WORD BAD_DT-10$ : 22 ZI
0202' 07BA 669 .WORD BAD_DT-10$ : 23 ZEM
0202' 07BC 670 .WORD BAD_DT-10$ : 24 DSC
0202' 07BE 671 .WORD BAD_DT-10$ : 25 OU
0202' 07C0 672 .WORD BAD_DT-10$ : 26 O
0202' 07C2 673 .WORD BAD_DT-10$ : 27 G
0202' 07C4 674 .WORD BAD_DT-10$ : 28 H
0202' 07C6 675 .WORD BAD_DT-10$ : 29 GC
0202' 07C8 676 .WORD BAD_DT-10$ : 30 HC
0103' 07CA 677 .WORD 80$-10$ : 31 COBOL intermediate data type
01BF 31 07CC 678 BRW BAD_DT
      07CF 679
      07CF 680 :+
      07CF 681 : CIT overflowed.
      07CF 682 :-
      07CF 683 3$:
00000000'8F DD 07CF 684 PUSHL #COB$_INTXPOVE ; Overflow signal
06 11 07D5 685 BRB 6$ ; go signal
      07D7 686
      07D7 687 :+
      07D7 688 : CIT underflow
      07D7 689 :-
      07D7 690 5$:
00000000'8F DD 07D7 691 PUSHL #COB$_INTXPUND ; Underflow signal
00000000'GF 01 FB 07DD 692 6$: CALLS #1,G^CIB$STOP ; Signal and stop.
      07E4 693
      07E4 694 :+
      07E4 695 : Destination is W
      07E4 696 :-
09 03 56 D4 07E4 697 20$: CLRL R6 ; Assume class S
      07E6 698 CMPB DSC$_CLASS(R0),#DSC$_CLASS_SD
      07EA 699 BNEQ 21$ ; Branch if not class SD
56 08 A0 98 07EC 700 CVTBL DSC$_SCALE(R0),R6 ; Get scale factor
      56 56 CE 07F0 701 MNEGL R6,R6 ; Negate scale factor
57 04 AE 9E 07F3 702 21$: MOVAB 4(SP),R7 ; Get source address
58 04 A0 D0 07F7 703 MOVL DSC$_POINTER(R0),R8 ; Get destination address
00000000'GF 16 07FB 704 JSB G^COB$_CVTIW_R8 ; Go to conversion routine
50 8E D0 0801 705 MOVL (SP)+,R0 ; Restore status
      04 0804 706 RET ; Return
      0805 707
      0805 708 :+
      0805 709 : Destination is L
      0805 710 :-
09 03 56 D4 0805 711 30$: CLRL R6 ; Assume class S
      0807 712 CMPB DSC$_CLASS(R0),#DSC$_CLASS_SD
      080B 713 BNEQ 31$ ; Branch if not class SD
56 08 A0 98 080D 714 CVTBL DSC$_SCALE(R0),R6 ; Get scale factor
      56 56 CE 0811 715 MNEGL R6,R6 ; Negate scale factor
57 04 AE 9E 0814 716 31$: MOVAB 4(SP),R7 ; Get source address
58 04 A0 D0 0818 717 MOVL DSC$_POINTER(R0),R8 ; Get destination address
00000000'GF 16 081C 718 JSB G^COB$_CVTIL_R8 ; Go to conversion routine
50 8E D0 0822 719 MOVL (SP)+,R0 ; Restore status
      04 0825 720 RET ; Return
      0826 721
      0826 722 :+
      0826 723 : Destination is Q
      0826 724 :-
```

```
09 03 56 D4 0826 725 40$: CLRL R6 ; Assume class S
      AO 91 0828 726      CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD
      07 12 082C 727      BNEQ 41$ ; Branch if not class SD
56 08 AO 98 082E 728      CVTBL DSC$B_SCALE(R0),R6 ; Get negative of scale factor
      56 56 CE 0832 729      MNEGL R6,R6 ;
57 04 AE 9E 0835 730 41$: MOVAB 4(SP),R7 ; Get source address
      AO DO 0839 731      MOVL DSC$A_POINTER(R0),R8 ; Get destination address
00000000 GF 16 083D 732      JSB G^COB$CVTIQ_R8 ; Go to conversion routine
      50 8E DO 0843 733      MOVL (SP)+,R0 ; Restore status
      04 0846 734      RET ; Return
      0847 735
      0847 736 ;+
      0847 737 ; Destination is F
      0847 738 ;-
56 04 AE 9E 0847 739 50$: MOVAB 4(SP),R6 ; Get source address
      AO DO 084B 740      MOVL DSC$A_POINTER(R0),R7 ; Get destination address
00000000 GF 16 084F 741      JSB G^COB$CVTIF_R7 ; Go to conversion routine
      50 8E DO 0855 742      MOVL (SP)+,R0 ; Restore status
      04 0858 743      RET ; Return
      0859 744
      0859 745 ;+
      0859 746 ; Destination is D
      0859 747 ;-
56 04 AE 9E 0859 748 60$: MOVAB 4(SP),R6 ; Get source address
      AO DO 085D 749      MOVL DSC$A_POINTER(R0),R7 ; Get destination address
00000000 GF 16 0861 750      JSB G^COB$CVTID_R7 ; Go to conversion routine
      50 8E DO 0867 751      MOVL (SP)+,R0 ; Restore status
      04 086A 752      RET ; Return
      086B 753
      086B 754 ;+
      086B 755 ; Destination is P
      086B 756 ;-
09 03 56 D4 086B 757 70$: CLRL R6 ; Assume class S
      AO 91 086D 758      CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD
      07 12 0871 759      BNEQ 71$ ; Branch if not class SD
56 08 AO 98 0873 760      CVTBL DSC$B_SCALE(R0),R6 ; Get negative of scale factor
      56 56 CE 0877 761      MNEGL R6,R6 ;
57 04 AE 9E 087A 762 71$: MOVAB 4(SP),R7 ; Get source address
      AO 3C 087E 763      MOVZWL DSC$W_LENGTH(R0),R8 ; Get destination length
59 04 AO DO 0881 764      MOVL DSC$A_POINTER(R0),R9 ; Get destination address
00000000 GF 16 0885 765      JSB G^COB$CVTIP_R9 ; Go to conversion routine
      50 8E DO 088B 766      MOVL (SP)+,R0 ; Restore status
      04 088E 767      RET ; Return
      088F 768
      088F 769 ;+
      088F 770 ; Destination is intermediate
      088F 771 ;-
50 04 AO DO 088F 772 80$: MOVL DSC$A_POINTER(R0),R0 ; Get destination address
80 04 AE 7D 0893 773      MOVQ 4(SP),(R0)+ ; Move 8 bytes
60 0C AE DO 0897 774      MOVL 12(SP),(R0) ; Move 4 more bytes
      50 8E DO 089B 775      MOVL (SP)+,R0 ; Restore status
      04 089E 776      RET ; Return
      089F 777 ;
```

```
089F 779 .SBTTL CONVERT Internal routine to convert to intermediate
089F 780
089F 781 :+
089F 782 :
089F 783 :
089F 784 :
089F 785 :-
089F 786
089F 787 CONVERT:
1F 00 02 A0 8F 089F 788 CASEB DSC$B_DTYPE(R0),#0,#31 ; Go to proper conversion code
00EA' 08A4 789 10$: .WORD BAD_DT-10$ : 0 Z
00EA' 08A6 790 .WORD BAD_DT-10$ : 1 V
00EA' 08A8 791 .WORD BAD_DT-10$ : 2 BU
00EA' 08AA 792 .WORD BAD_DT-10$ : 3 WU
00EA' 08AC 793 .WORD BAD_DT-10$ : 4 LU
00EA' 08AE 794 .WORD BAD_DT-10$ : 5 QU
00EA' 08B0 795 .WORD BAD_DT-10$ : 6 B
0043' 08B2 796 .WORD 20$-10$ : 7 W
005C' 08B4 797 .WORD 30$-10$ : 8 L
0075' 08B6 798 .WORD 40$-10$ : 9 Q
008E' 08B8 799 .WORD 50$-10$ : 10 F
009B' 08BA 800 .WORD 60$-10$ : 11 D
00EA' 08BC 801 .WORD BAD_DT-10$ : 12 FC
00EA' 08BE 802 .WORD BAD_DT-10$ : 13 DC
00EA' 08C0 803 .WORD BAD_DT-10$ : 14 T
00EA' 08C2 804 .WORD BAD_DT-10$ : 15 NU
00EA' 08C4 805 .WORD BAD_DT-10$ : 16 NL
00EA' 08C6 806 .WORD BAD_DT-10$ : 17 NLO
00EA' 08C8 807 .WORD BAD_DT-10$ : 18 NR
00EA' 08CA 808 .WORD BAD_DT-10$ : 19 NRO
00EA' 08CC 809 .WORD BAD_DT-10$ : 20 NZ
00A8' 08CE 810 .WORD 70$-10$ : 21 P
00EA' 08D0 811 .WORD BAD_DT-10$ : 22 ZI
00EA' 08D2 812 .WORD BAD_DT-10$ : 23 ZEM
00EA' 08D4 813 .WORD BAD_DT-10$ : 24 DSC
00EA' 08D6 814 .WORD BAD_DT-10$ : 25 OU
00EA' 08D8 815 .WORD BAD_DT-10$ : 26 O
00EA' 08DA 816 .WORD BAD_DT-10$ : 27 G
00EA' 08DC 817 .WORD BAD_DT-10$ : 28 H
00EA' 08DE 818 .WORD BAD_DT-10$ : 29 GC
00EA' 08E0 819 .WORD BAD_DT-10$ : 30 HC
00C4' 08E2 820 .WORD 80$-10$ : 31 COBOL intermediate data type
00A7 31 08E4 821 BRW BAD_DT
08E7 822
08E7 823 :+
08E7 824 : Source is W
08E7 825 :-
08E7 826 20$: CLRL R6 ; Assume class S
09 03 A0 91 08E9 827 CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD ; Branch if not class SD
04 12 08ED 828 BNEQ 21$ ; Get scale factor
56 08 A0 98 08EF 829 CVTBL DSC$B_SCALE(R0),R6 ; Get source address
57 04 A0 D0 08F3 830 21$: MOVL DSC$A_POINTER(R0),R7 ; Get destination address
58 51 D0 08F7 831 MOVL R1,R8 ; Go to conversion routine
00000000'GF 17 08FA 832 JMP G^COB$CVTWI_R8
0900 833
0900 834 :+
0900 835 : Source is L
```

```
09 03 56 D4 0900 836 :- CLRL R6 ; Assume class S
04 91 0902 837 30$: CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD
08 12 0906 838 BNEQ 31$ ; Branch if not class SD
56 08 A0 98 0908 840 CVTBL DSC$B_SCALE(R0),R6 ; Get scale factor
57 04 A0 D0 090C 841 31$: MOVL DSC$A_POINTER(R0),R7 ; Get source address
58 51 D0 0910 842 MOVL R1,R8 ; Get destination address
00000000'GF 17 0913 843 JMP G^COB$CVTLI_R8 ; Go to conversion routine
0919 844
0919 845 :+
0919 846 : Source is Q
0919 847 :-
09 03 56 D4 0919 848 40$: CLRL R6 ; Assume class S
04 91 091B 849 CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD
08 12 091F 850 BNEQ 41$ ; Branch if not class SD
56 08 A0 98 0921 851 CVTBL DSC$B_SCALE(R0),R6 ; Get scale factor
57 04 A0 D0 0925 852 41$: MOVL DSC$A_POINTER(R0),R7 ; Get source address
58 51 D0 0929 853 MOVL R1,R8 ; Get destination address
00000000'GF 17 092C 854 JMP G^COB$CVTQI_R8 ; Go to conversion routine
0932 855
0932 856 :+
0932 857 : Source is F
0932 858 :-
56 04 A0 D0 0932 859 50$: MOVL DSC$A_POINTER(R0),R6 ; Get source address
57 51 D0 0936 860 MOVL R1,R7 ; Get destination address
00000000'GF 17 0939 861 JMP G^COB$CVTFI_R7 ; Go to conversion routine
093F 862
093F 863 :+
093F 864 : Source is D
093F 865 :-
56 04 A0 D0 093F 866 60$: MOVL DSC$A_POINTER(R0),R6 ; Get source address
57 51 D0 0943 867 MOVL R1,R7 ; Get destination address
00000000'GF 17 0946 868 JMP G^COB$CVTDI_R7 ; Go to conversion routine
094C 869
094C 870 :+
094C 871 : Source is P
094C 872 :-
09 03 56 D4 094C 873 70$: CLRL R6 ; Assume class S
04 91 094E 874 CMPB DSC$B_CLASS(R0),#DSC$K_CLASS_SD
08 12 0952 875 BNEQ 71$ ; Branch if not class SD
56 08 A0 98 0954 876 CVTBL DSC$B_SCALE(R0),R6 ; Get scale factor
57 60 3C 0958 877 71$: MOVZWL DSC$W_LENGTH(R0),R7 ; Get source length
58 04 A0 D0 095B 878 MOVL DSC$A_POINTER(R0),R8 ; Get source address
59 51 D0 095F 879 MOVL R1,R9 ; Get destination address
00000000'GF 17 0962 880 JMP G^COB$CVTPI_R9 ; Go to conversion routine
0968 881
0968 882 :+
0968 883 : Source is intermediate
0968 884 :-
50 04 A0 D0 0968 885 80$: MOVL DSC$A_POINTER(R0),R0 ; Get source address
0063 8F 60 B1 096C 886 CMPW INT$W_I_EXP(R0), #INT$K_I_EXP_HI ; Bigger than max ?
0E 14 0971 887 BGTR 81$ ; Yes, overflow
FF9D 8F 60 B1 0973 888 CMPW INT$W_I_EXP(R0), #INT$K_I_EXP_LO ; Less than min ?
07 19 0978 889 BLSS 81$ ; Yes, underflow
81 80 7D 097A 890 MOVQ (R0)+,(R1)+ ; Copy 8 bytes
61 60 D0 097D 891 MOVL (R0),(R1) ; Copy 4 more bytes
05 0980 892 RSB ; Done
```

COB\$EXPI
1-012

COBOL Intermediate Exponentiate
CONVERT Internal routine to convert to

B 16

15-SEP-1984 23:43:25 VAX/VMS Macro V04-00
6-SEP-1984 10:45:28 [COBRTL.SRC]COBEXPI.MAR;1

Page 21
(10)

```
00000000'8F DD 0981 893 81$:  PUSHL  #COB$ INTRESOPE      ; Intermediate reserved operand
00000000'GF  01  FB 0987 894      CALLS  #1,G^[IB$STOP      ; Signal the error
                   098E 895
                   098E 896 :+
                   098E 897 : Here if not a supported data type.
                   098E 898 :-
00000000'8F DD 098E 899 BAD_DT:  PUSHL  #COB$ INVARG        ; "Invalid argument list"
00000000'GF  01  FB 0994 900      CALLS  #1,G^[IB$STOP
                   099B 901
                   099B 902      .END
```

COB\$EXPI
Symbol table

COBOL Intermediate Exponentiate

C 16

15-SEP-1984 23:43:25 VAX/VMS Macro V04-00
6-SEP-1984 10:45:28 [COBRTL.SRC]COB\$EXPI.MAR;1

Page 22
(10)

AS	= 00000000		
BS	= 00000012		
BAD_DT	0000098E	R	02
BASE_IT	= 00000000		
CS	= 00000024		
COB\$CVTDI-R7	*****	X	00
COB\$CVTFI-R7	*****	X	00
COB\$CVTID-R7	*****	X	00
COB\$CVTIF-R7	*****	X	00
COB\$CVTIL-R8	*****	X	00
COB\$CVTIP-R9	*****	X	00
COB\$CVTIQ-R8	*****	X	00
COB\$CVTIW-R8	*****	X	00
COB\$CVTLI-R8	*****	X	00
COB\$CVTPI-R9	*****	X	00
COB\$CVTQI-R8	*****	X	00
COB\$CVTWI-R8	*****	X	00
COB\$EXPI	00000550	RG	02
COB\$EXPI_OSE	00000549	RG	02
COB\$_INT\$EXPOVE	*****	X	00
COB\$_INT\$EXPUND	*****	X	00
COB\$_INT\$RESOPE	*****	X	00
COB\$_INVARG	*****	X	00
COB\$_UNDEF_EXP	*****	X	00
CONVERT	0000089F	R	02
DS	= 00000030		
DSC\$A_POINTER	= 00000004		
DSC\$B_CLASS	= 00000003		
DSC\$B_DTYPE	= 00000002		
DSC\$B_SCALE	= 00000008		
DSC\$K_CLASS_SD	= 00000009		
DSC\$W_LENGTH	= 00000000		
ES	= 0000003C		
ERR_0	00000525	R	02
ERR_BAD	00000535	R	02
EXP_IT	= 0000000C		
EXP_J	00000554	R	02
E_D	= 00000002		
FINISH	0000076A	R	02
INT\$K_I_EXP_HI	= 00000063		
INT\$K_I_EXP_LO	= FFFFFFF9D		
INT\$K_I_FRACT_D	= 00000012		
INT\$K_I_LEN	= 0000000C		
INT\$P_I_FRACT	= 00000002		
INT\$W_I_EXP	= 00000000		
LIB\$STOP	*****	X	00
MULBIG	00000402	R	02
NINES	000003F2	R	02
OFFSET	= 00000066		
OSE	= 00000065		
P0	000003F0	R	02
P1	000003F1	R	02
RES_IT	= 00000018		
RES_SIGN	= 00000064		
TABLE1	= FFFFFFFF0	R	02
TABLE2	= 000001F0	R	02
X	= 00000024		

= 00000034
= 00000044

Y
Z

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR
_COB\$CODE	0000099B (2459.)	02 (2.)	PIC USR

CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.04	00:00:01.74
Command processing	123	00:00:00.41	00:00:03.53
Pass 1	190	00:00:02.87	00:00:13.45
Symbol table sort	0	00:00:00.17	00:00:00.85
Pass 2	174	00:00:01.43	00:00:04.60
Symbol table output	9	00:00:00.04	00:00:00.06
Psect synopsis output	3	00:00:00.02	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	534	00:00:04.98	00:00:24.28

The working set limit was 1350 pages.
27182 bytes (54 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 189 non-local and 46 local symbols.
902 source lines were read in Pass 1, producing 22 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[COBRTL.OBJ]COBRTL.MLB;1	1
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

203 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LISS:COBEXPI/OBJ=OBJ\$:COBEXPI MSRC\$:COBEXPI/UPDATE=(ENH\$:COBEXPI)+LIB\$:COBRT

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

