

[illegible]

3-

Sy

LI

LI
LILI
LI
LILI
LI

LI

LI
LILI
LILI
LI

LI

LI
LI

LI

LI
LI

LI

LI
LI

LI

11

LI

LI
LI

LI

LI
LI

21

LI

LI
LI

22

11

LI
LI

LI

LI
LI

LI

OT:
1-

```

000000      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      VV      VV      TTTTTTTTTT      LL      TTTTTTTTTT
000000      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      VV      VV      TTTTTTTTTT      LL      TTTTTTTTTT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
00          TT          SS          CC          VV      VV      TT          LL          TT
000000      TT          SSSSSSSS      CCCCCCCC      VV      VV      TT          LLLLLLLLLL      TT
000000      TT          SSSSSSSS      CCCCCCCC      VV      VV      TT          LLLLLLLLLL      TT
                                ...
                                ...
                                ...
                                ...

LL          IIIIIIII      SSSSSSSS
LL          IIIIIIII      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
LLLLLLLLLLL IIIIIIII      SSSSSSSS
LLLLLLLLLLL IIIIIIII      SSSSSSSS

```


(2)	45	Edit History
(3)	67	DECLARATIONS
(4)	102	OTSSCVT_L_T0 - Long to text, 0 format
(5)	227	OTSSCVT_L_TZ - Long to text, Z format
(6)	361	OTSSCVT_L_TB - Long to text, binary format
(7)	449	Local subroutines
(8)	522	OTSSCVT_L_TI - Long to text, I format
(9)	690	OTSSCVT_L_TU - Long to text, unsigned decimal format
(10)	782	OTSSCVT_L_TL - Long to text, L format

```

0000 1      .TITLE OTSSCVTLT - Convert longword to text, 0, Z, L, B, U, I formats
0000 2      .IDENT /1-014/                               ; File: OTSCVTLT.MAR Edit: MDL1014
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10 *
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17 *
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21 *
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: Language independent support library
0000 31
0000 32 ABSTRACT:
0000 33
0000 34 Routines to convert values of any length to text using 0 (octal),
0000 35 Z (hexadecimal), B (binary) and L (logical) formats. Also routines to
0000 36 convert byte, word and longword integers to text using I (signed decimal)
0000 37 and unsigned decimal formats.
0000 38
0000 39 ENVIRONMENT: User Mode, AST Reentrant
0000 40
0000 41 --
0000 42 AUTHOR: Steven B. Lionel, CREATION DATE: 21-Mar-1979
0000 43

```



```

0000 45      .SBTTL Edit History
0000 46      :
0000 47      : 1-001 - Original. Replaces FOR$CNVOI. SBL 22-Mar-1979
0000 48      : 1-002 - Move V FORCEPLUS to bit position 1. SBL 25-July-1979
0000 49      : 1-003 - Speed improvements. New I format logic. Use CASEB in
0000 50      : INITIALIZE. SBL 28-Dec-1979
0000 51      : 1-004 - Fix bug in CASE. SBL 31-Dec-1979
0000 52      : 1-005 - Do correct thing for >128 arguments. SBL 31-Dec-1979
0000 53      : 1-006 - Minor code improvements courtesy of Rich Grove. SBL 2-Jan-1980
0000 54      : 1-007 - More minor code improvements. Make value_size of zero an error.
0000 55      : SBL 3-Jan-1980
0000 56      : 1-008 - Fix bug where OTSSCVT_L TI of 3 arguments doesnt fetch
0000 57      : the value. SBL 13-March-1980
0000 58      : 1-009 - Add OTSSCVT_L TB. SBL 6-Nov-1980
0000 59      : 1-010 - Make OTSSCVT_L TI produce a blank field when value is zero and
0000 60      : int_digits is zero. SPR 11-37827 JAW 22-May-1981
0000 61      : 1-011 - Add bit_offset and flags parameter to B, O and Z conversions. SBL 6-July-
0000 62      : 1-012 - Reverse order of bit_offset and flags parameters in B, O and Z. SBL 30-Dec-
0000 63      : 1-013 - Add OTSSCVT_L TU. SBL 27-Apr-1983
0000 64      : 1-014 - fix bug where OTSSCVT_L TU of 4 or 5 arguments doesnt fetch the
0000 65      : value. MDL 25-May-1984

```

```

0000 67      .SBTTL DECLARATIONS
0000 68 :
0000 69 : INCLUDE FILES:
0000 70 :
0000 71 :
0000 72 :
0000 73 : EXTERNAL DECLARATIONS:
0000 74 :
0000 75      .DSABL GBL                                : Prevent undeclared
0000 76                                         : symbols from being
0000 77                                         : automatically global.
0000 78      .EXTRN OTSS_OUTCONERR                    : error code
0000 79
0000 80 :
0000 81 : MACROS:
0000 82 :
0000 83 :
0000 84 :
0000 85 : EQUATED SYMBOLS:
0000 86 :
0000 87 :
0000 88 :
0000 89 : PSECT DECLARATIONS:
0000 90 :
0000 91      .PSECT _OTSS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 92      EXE, RD, NOWRT, LONG
0000 93
0000 94 :
0000 95 : OWN STORAGE:
0000 96 :
0000 97 :
0000 98 LETTERS:
0000 99      .ASCII /0123456789ABCDEF/           ; Characters for output
000C
0010 100

```

42 41 39 38 37 36 35 34 33 32 31 30
46 45 44 43


```
0010 102 .SBTTL OTSS$CVT_L_TO - Long to text, 0 format
0010 103 :++
0010 104 : FUNCTIONAL DESCRIPTION:
0010 105 :
0010 106 : This routine converts its input value to a text representation,
0010 107 : using base 8 (octal). The input value may be of any length.
0010 108 :
0010 109 : OTSS$CVT_L_TO supports FORTRAN 0w and 0w.m output conversion.
0010 110 :
0010 111 : A separate entry point FOR$CNV_OUT_0 is provided for compatibility
0010 112 : with previous releases. Note that the input value for
0010 113 : OTSS$CVT_L_TO is by reference while that for FOR$CNV_OUT_0 is
0010 114 : by value.
0010 115 :
0010 116 : CALLING SEQUENCE:
0010 117 :
0010 118 : status.wlc.v = OTSS$CVT_L_TO (value.rx.r, out_string.wt.ds
0010 119 : [ , int_digits.rl.v
0010 120 : [ , value_size.rl.v
0010 121 : [ , flags.rbu.v
0010 122 : [ , bit_offset.rl.v]]])
0010 123 :
0010 124 : status.wlc.v = FOR$CNV_OUT_0 (value.rl.v, out_string.wt.ds)
0010 125 :
0010 126 :
0010 127 : INPUT PARAMETERS:
0010 128 :
00000004 0010 129 : value = 4 ; Input value to be converted to text
0000000C 0010 130 : int_digits = 12 ; Minimum number of digits to be produced.
0010 131 : ; If actual number of significant digits
0010 132 : ; is smaller, leading zeroes will be
0010 133 : ; produced. If int_digits is zero
0010 134 : ; and value is zero, a blank field will
0010 135 : ; result. The default is 1.
00000010 0010 136 : value_size = 16 ; The size of value in bytes. The
0010 137 : ; default is 4 if this argument is not present.
0010 138 : ; If V_SIZE_IN_BITS set, value_size is in units of b
00000014 0010 139 : flags = 20 ; Caller supplied flags
00000002 0010 140 : V_SIZE_IN_BITS = 2 ; "value_size" is in bits rather than bytes.
00000018 0010 141 : bit_offset = 24 ; Starting bit position. Default is 0.
0010 142 :
0010 143 :
0010 144 : IMPLICIT INPUTS:
0010 145 :
0010 146 : NONE
0010 147 :
0010 148 : OUTPUT PARAMETERS:
0010 149 :
00000008 0010 150 : out_string = 8 ; Output string by descriptor.
0010 151 :
0010 152 : IMPLICIT OUTPUTS:
0010 153 :
0010 154 : NONE
0010 155 :
0010 156 : COMPLETION CODES:
0010 157 :
0010 158 : $$$_NORMAL - Successful completion.
```



```
0010 159 : OTSS_OUTCONERR - Output conversion error. The converted value
0010 160 : does not fit in the field provided. The field
0010 161 : is filled with asterisks. This error is also
0010 162 : given if value_size is not positive.
0010 163 :
0010 164 : SIDE EFFECTS:
0010 165 :
0010 166 : NONE
0010 167 :
0010 168 : --
0010 169 :
00FC 0010 170 .ENTRY FOR$CNV_OUT_0, ^M<R2,R3,R4,R5,R6,R7>
0012 171
54 04 AC 9E 0012 172 MOVAB value(AP), R4 ; Address of value
015B 30 0016 173 BSBW INITIALIZE ; Set up default values
53 20 D0 0019 174 MOVL #32, R3 ; Value MUST be 4 bytes!
09 11 001C 175 BRB COMMON_0 ; Go to common routine
001E 176
00FC 001E 177 .ENTRY OTSSCVT_L_TO , ^M<R2,R3,R4,R5,R6,R7>
0020 178
54 04 AC D0 0020 179 MOVL value(AP), R4 ; Address of value
014D 30 0024 180 BSBW INITIALIZE ; Set up default values
0027 181
0027 182
0027 183 COMMON_0:
0027 184 MOVL R1, R6 ; last set char address
01 01 53 002A 185 10$: CASEL R3, #1, #1 ; Select on bits remaining
0006' 002E 186 1$: .WORD 11$-1$ ; 1 bit
001B' 0030 187 .WORD 12$-1$ ; 2 bits
2A 11 0032 188 BRB 13$ ; 3 or more bits
0034 189 ; can't be zero
57 64 01 55 EF 0034 190 11$: EXTZV R5, #1, (R4), R7 ; extract one bit
3E 13 0039 191 BEQL EXIT_0 ; if zero, exit
0176 30 003B 192 BSBW ZERO_FILL ; fill with zero to this point
71 BE AF47 90 003E 193 MOVB LETTERS[R7], -(R1) ; move character
52 D7 0043 194 DECL R2 ; decrement digits count
56 D7 0045 195 DECL R6 ; decrement place holder
30 11 0047 196 BRB EXIT_0
57 64 02 55 EF 0049 197 12$: EXTZV R5, #2, (R4), R7 ; extract 2 bits
29 15 004E 198 BLEQ EXIT_0 ; if zero, finish
0161 30 0050 199 BSBW ZERO_FILL ; fill with zeroes
71 A9 AF47 90 0053 200 MOVB LETTERS[R7], -(R1) ; move character
52 D7 0058 201 DECL R2 ; decrement digits count
56 D7 005A 202 DECL R6 ; decrement place holder
1B 11 005C 203 BRB EXIT_0 ; exit
57 64 03 55 EF 005E 204 13$: EXTZV R5, #3, (R4), R7 ; extract 3 bits
0A 13 0063 205 BEQL 40$ ; skip insert if zero
014C 30 0065 206 BSBW ZERO_FILL ; fill with zeroes
76 94 AF47 90 0068 207 MOVB LETTERS[R7], -(R6) ; move character
52 D7 006D 208 DECL R2 ; decrement digits count
55 03 C0 006F 209 40$: ADDL2 #3, R5 ; increment position
51 D7 0072 210 DECL R1 ; decrement character pointer
53 03 C2 0074 211 SUBL2 #3, R3 ; decrement count
B1 1A 0077 212 BGTRU 10$ ; continue if not done
0079 213
0079 214 EXIT_0:
52 D5 0079 215 TSTL R2 ; more zeroes to fill?
```


51	56	09	15	007B	216	BLEQ	50\$	:	no
		52	C3	007D	217	SUBL3	R2, R6, R1	:	insert R2 zeroes
		50	D6	0081	218	INCL	R0	:	we aren't writing a char here
	012E	30	0083	219	BSBW	ZERO_FILL		:	fill with zeroes
		50	D5	0086	220	TSTL	R0	:	Blank fill?
		06	15	0088	221	BLEQ	70\$	:	No
	76	20	90	008A	222	MOVB	#^A/ / -(R6)	:	Move a blank
	FA	50	F5	008D	223	SOBGTR	R0, 60\$	:	Loop till done
	50	01	D0	0090	224	MOVL	#1, R0	:	SS\$ NORMAL
			04	0093	225	RET		:	exit

```
0094 227 .SBTTL OTSSCVT_L_TZ - Long to text, Z format
0094 228
0094 229 : FUNCTIONAL DESCRIPTION:
0094 230 :
0094 231 : This routine converts its input value to a text representation,
0094 232 : using base 16 (hexadecimal). The input value may be of any length.
0094 233 :
0094 234 : OTSSCVT_L_TZ supports FORTRAN Zw and Zw.m output conversion.
0094 235 :
0094 236 : A separate entry point FOR$CNV_OUT_Z is provided for compatibility
0094 237 : with previous releases. Note that the input value for
0094 238 : OTSSCVT_L_TZ is by reference while that for FOR$CNV_OUT_Z is
0094 239 : by value.
0094 240 :
0094 241 : CALLING SEQUENCE:
0094 242 :
0094 243 : status.wlc.v = OTSSCVT_L_TZ (value.rx.r, out_string.wt.ds
0094 244 :                               [, int_digits.rl.v
0094 245 :                               [, value_size.rl.v
0094 246 :                               [, flags.rbu.v
0094 247 :                               [, bit_offset.rl.v]]])
0094 248 :
0094 249 : status.wlc.v = FOR$CNV_OUT_Z (value.rl.v, out_string.wt.ds)
0094 250 :
0094 251 : INPUT PARAMETERS:
0094 252 :
0094 253 :
00000004 0094 254 : value = 4 ; Input value to be converted to text
0000000C 0094 255 : int_digits = 12 ; Minimum number of digits to be produced.
0094 256 : ; If actual number of significant digits
0094 257 : ; is smaller, leading zeroes will be
0094 258 : ; produced. If int_digits is zero
0094 259 : ; and value is zero, a blank field will
0094 260 : ; result. The default is 1.
00000010 0094 261 : value_size = 16 ; The size of value in bytes. The
0094 262 : ; default is 4 if this argument is not present.
0094 263 : ; If flags bit V_SIZE_IN_BITS is set, value_size
0094 264 : ; is the number of bits in the value.
00000014 0094 265 : flags = 20 ; Caller supplied flags. Defined bits are:
00000002 0094 266 : V_SIZE_IN_BITS = 2 ; "value_size" is in units of bits
00000018 0094 267 : bit_offset = 24 ; Offset of value in bits. Default is zero.
0094 268 :
0094 269 : IMPLICIT INPUTS:
0094 270 :
0094 271 :
0094 272 : NONE
0094 273 :
0094 274 : OUTPUT PARAMETERS:
0094 275 :
00000008 0094 276 : out_string = 8 ; Output string by descriptor.
0094 277 :
0094 278 : IMPLICIT OUTPUTS:
0094 279 :
0094 280 : NONE
0094 281 :
0094 282 : COMPLETION CODES:
0094 283 :
```



```
0094 284 : SSS NORMAL - Successful completion.
0094 285 : OTSS_OUTCONERR - Output conversion error. The converted value
0094 286 : does not fit in the field provided. The field
0094 287 : is filled with asterisks. This error is also given
0094 288 : if value_size is not positive.
0094 289 :
0094 290 : SIDE EFFECTS:
0094 291 :
0094 292 : NONE
0094 293 :
0094 294 : --
0094 295 :
00FC 0094 296 .ENTRY FOR$CNV_OUT_Z, ^M<R2,R3,R4,R5,R6,R7>
0096 297
54 04 AC 9E 0096 298 MOVAB value(AP), R4 ; Address of value
00D7 30 009A 299 BSBW INITIALIZE ; Set up default values
53 20 D0 009D 300 MOVL #32, R3 ; Value MUST be 4 bytes!
09 11 00A0 301 BRB COMMON_Z ; Go to common routine
00A2 302
00FC 00A2 303 .ENTRY OTSS$CVT_L_TZ, ^M<R2,R3,R4,R5,R6,R7>
00A4 304
54 04 AC D0 00A4 305 MOVL value(AP), R4 ; Address of value
00C9 30 00A8 306 BSBW INITIALIZE ; Set up default values
00AB 307
00AB 308
00AB 309 COMMON_Z:
02 56 51 D0 00AB 310 MOVL R1, R6 ; last set char address
01 53 CF 00AE 311 10$: CASEL R3, #1, #2 ; Select on bits remaining
0008' 00B2 312 1$: .WORD 11$-1$ ; 1 bit
001E' 00B4 313 .WORD 12$-1$ ; 2 bits
0034' 00B6 314 .WORD 13$-1$ ; 3 bits
42 11 00B8 315 BRB 14$ ; 4 or more bits
00BA 316 ; can't be zero
57 64 01 55 EF 00BA 317 11$: EXTZV R5, #1, (R4), R7 ; extract one bit
57 13 00BF 318 BEQL EXIT_Z ; if zero, exit
00F0 30 00C1 319 BSBW ZERO_FILL ; fill with zero to this point
71 FF37 CF47 90 00C4 320 MOVB LETTERS[R7], -(R1) ; move character
52 D7 00CA 321 DECL R2 ; decrement digits count
56 D7 00CC 322 DECL R6 ; decrement place holder
48 11 00CE 323 BRB EXIT_Z
57 64 02 55 EF 00D0 324 12$: EXTZV R5, #2, (R4), R7 ; extract 2 bits
41 15 00D5 325 BLEQ EXIT_Z ; if zero, finish
00DA 30 00D7 326 BSBW ZERO_FILL ; fill with zeroes
71 FF21 CF47 90 00DA 327 MOVB LETTERS[R7], -(R1) ; move character
52 D7 00E0 328 DECL R2 ; decrement digits count
56 D7 00E2 329 DECL R6 ; decrement place holder
32 11 00E4 330 BRB EXIT_Z ; exit
57 64 03 55 EF 00E6 331 13$: EXTZV R5, #3, (R4), R7 ; extract 3 bits
2B 13 00EB 332 BEQL EXIT_Z ; skip insert if zero
00C4 30 00ED 333 BSBW ZERO_FILL ; fill with zeroes
71 FF0B CF47 90 00F0 334 MOVB LETTERS[R7], -(R1) ; move character
52 D7 00F6 335 DECL R2 ; decrement digits count
56 D7 00F8 336 DECL R6 ; decrement place holder
1C 11 00FA 337 BRB EXIT_Z
57 64 04 55 EF 00FC 338 14$: EXTZV R5, #4, (R4), R7 ; extract 4 bits
0B 13 0101 339 BEQL 40$ ; skip insert if zero
00AE 30 0103 340 BSBW ZERO_FILL ; fill with zeroes
```

PC	Instruction	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
----	-------------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I


```

0133 361 .SBTTL OTSS$CVT_L_TB - Long to text, binary format
0133 362 :++
0133 363 : FUNCTIONAL DESCRIPTION:
0133 364 :
0133 365 : This routine converts its input value to a text representation,
0133 366 : using base 2 (binary). The input value may be of any length.
0133 367 :
0133 368 : CALLING SEQUENCE:
0133 369 :
0133 370 : status.wlc.v = OTSS$CVT_L_TB (value.rx.r, out_string.wt.ds
0133 371 : [, int_digits.rl.v
0133 372 : [, value_size.rl.v
0133 373 : [, flags.rbu.v
0133 374 : [, bit_offset.rl.v]]])
0133 375 :
0133 376 : INPUT PARAMETERS:
0133 377 :
00000004 0133 378 : value = 4 ; Input value to be converted to text
0000000C 0133 379 : int_digits = 12 ; Minimum number of digits to be produced.
0133 380 : ; If actual number of significant digits
0133 381 : ; is smaller, leading zeroes will be
0133 382 : ; produced. If int_digits is zero
0133 383 : ; and value is zero, a blank field will
0133 384 : ; result. The default is 1.
00000010 0133 385 : value_size = 16 ; The size of value in bytes. The
0133 386 : ; default is 4 if this argument is not present.
0133 387 : ; If flags bit V_SIZE_IN_BITS is set, value_size
0133 388 : ; is the number of bits in the value.
00000014 0133 389 : flags = 20 ; Caller supplied flags. Defined bits are:
00000002 0133 390 : V_SIZE_IN_BITS = 2 ; "value_size" is in units of bits
00000018 0133 391 : bit_offset = 24 ; Offset of value in bits. Default is zero
0133 392 :
0133 393 :
0133 394 : IMPLICIT INPUTS:
0133 395 :
0133 396 : NONE
0133 397 :
0133 398 : OUTPUT PARAMETERS:
0133 399 :
00000008 0133 400 : out_string = 8 ; Output string by descriptor.
0133 401 :
0133 402 : IMPLICIT OUTPUTS:
0133 403 :
0133 404 : NONE
0133 405 :
0133 406 : COMPLETION CODES:
0133 407 :
0133 408 : SSS_NORMAL - Successful completion.
0133 409 : OTSS$_OUTCONERR - Output conversion error. The converted value
0133 410 : ; does not fit in the field provided. The field
0133 411 : ; is filled with asterisks. This error is also given
0133 412 : ; if value_size is not positive.
0133 413 :
0133 414 : SIDE EFFECTS:
0133 415 :
0133 416 : NONE
0133 417 :

```



```

                                0133 418 ;--
                                0133 419
                                0133 420 .ENTRY OTSS$CVT_L_TB , ^M<R2,R3,R4,R5,R6,R7>
                                0135 421
54 04 AC D0 0135 422 MOVL value(AP), R4 ; Address of value
0038 30 0139 423 BSBW INITIALIZE ; Set up default values
                                013C 424
57 64 56 51 D0 013C 425 MOVL R1, R6 ; Last set char address
01 55 EF 013F 426 10$: EXTZV R5, #1, (R4), R7 ; extract 1 bits
0B 13 0144 427 BEQL 20$ ; skip insert if zero
006B 30 0146 428 BSBW ZERO_FILL ; fill with zeroes
76 FEB2 CF47 90 0149 429 MOV B LETTERS[R7], -(R6) ; move character
52 D7 014F 430 DECL R2 ; decrement digits count
51 D7 0151 431 20$: DECL R1 ; decrement character pointer
55 D6 0153 432 INCL R5 ; increment bit position
53 D7 0155 433 DECL R3 ; decrement bit count
E6 14 0157 434 BGTR 10$ ; loop back if more bits
                                0159 435
                                0159 436 EXIT_B:
52 D5 0159 437 TSTL R2 ; more zeroes to fill?
09 15 015B 438 BLEQ 30$ ; no
51 56 52 C3 015D 439 SUBL3 R2, R6, R1 ; insert R2 zeroes
50 D6 0161 440 INCL R0 ; we aren't writing a char here
004E 30 0163 441 BSBW ZERO_FILL ; fill with zeroes
50 D5 0166 442 30$: TSTL R0 ; Blank fill?
06 15 0168 443 BLEQ 50$ ; No
76 20 90 016A 444 40$: MOV B #^A/ /, -(R6) ; Move a blank
FA 50 F5 016D 445 SOBGTR R0, 40$ ; Loop till done
50 01 D0 0170 446 50$: MOVL #1, R0 ; SS$ NORMAL
04 0173 447 RET ; exit
```



```

0174 449 .SBTTL Local subroutines
0174 450
0174 451 ;+
0174 452 :+ INITIALIZE - Perform common initialization
0174 453 :+
0174 454 :+ 1. R0 gets string length.
0174 455 :+ 2. R1 gets address of 1 byte past end of out_string.
0174 456 :+ 3. R2 gets int_digits value.
0174 457 :+ 4. R3 gets value size in bits.
0174 458 :+ 5. R5 gets starting bit position
0174 459 :-
0174 460
0174 461 INITIALIZE:
50 08 BC 7D 0174 462 MOVQ @out_string(AP), R0 ; Get string descriptor
50 50 50 3C 0178 463 MOVZWL R0, R0 ; R0 gets string length
51 50 C0 017B 464 ADDL2 R0, R1 ; R1 has 1 past last byte
52 01 D0 017E 465 MOVL #1, R2 ; default digits in int
53 20 D0 0181 466 MOVL #32, R3 ; default size in bits
55 55 D4 0184 467 CLRL R5 ; default bit offset
04 57 08 D0 0186 468 MOVL #8, R7 ; Default multiplier size-to-bits
04 02 6C 8F 0189 469 CASEB (AP), #2, #4 ; Select on argument count
0026' 018D 470 1$: .WORD 20$-1$ ; 2 arguments
0022' 018F 471 .WORD 30$-1$ ; 3 arguments
0016' 0191 472 .WORD 40$-1$ ; 4 arguments
000E' 0193 473 .WORD 50$-1$ ; 5 arguments
000A' 0195 474 .WORD 60$-1$ ; 6 arguments
0197 475 ; fall through ; Assume >6 arguments
55 18 AC D0 0197 476 60$: MOVL bit_offset(AP), R5 ; Get bit offset
03 14 AC 02 E1 019B 477 50$: BBC #V_SIZE_IN_BITS, flags(AP), 40$ ; Bit not set?
57 57 01 D0 01A0 478 MOVL #1, R7 ; value_size is in bits
53 57 10 AC C5 01A3 479 40$: MULL3 value_size(AP), R7, R3 ; Get size in bits
1D 1D 01A8 480 BVS ERROR ; Error if overflow
03 12 01AA 481 BNEQ 30$ ; Ok if not zero
53 20 D0 01AC 482 MOVL #32, R3 ; Assume 32 bits
52 0C AC D0 01AF 483 30$: MOVL int_digits(AP), R2 ; Get int_digits argument
05 05 01B3 484 20$: RSB ; End of initialization
01B4 485
01B4 486
01B4 487 ;+
01B4 488 :+ ZERO_FILL - Fill in skipped zeroes
01B4 489 :+
01B4 490 :+ ZERO_FILL is called whenever a main routine wishes to output
01B4 491 :+ a non-zero digit. First, it checks to see if there is room
01B4 492 :+ for one more character. If not, it branches to ERROR.
01B4 493 :+ It then fills with zeroes the space between the last non-zero
01B4 494 :+ digit and the current location, if any. It also updates the
01B4 495 :+ character pointers and counts appropriately.
01B4 496 :-
01B4 497
01B4 498 ZERO_FILL:
50 50 D7 01B4 499 DECL R0 ; Reduce char count
0F 19 01B6 500 BLSS ERROR ; If negative, out of room
56 51 D1 01B8 501 CMPL R1, R6 ; Any difference?
01 19 01BB 502 BLSS 10$ ; Yes
05 05 01BD 503 RSB ; No, exit
76 30 90 01BE 504 10$: MOVB #^A/O/, -(R6) ; Move a zero
52 52 D7 01C1 505 DECL R2 ; Decrement digits count

```



```

EF 11 01C3 506 BRB ZERO_FILL ; Loop till done
      01C5 507
      01C5 508 ;+
      01C5 509 ; ERROR - Return output conversion error
      01C5 510 ; Not used by L format.
      01C5 511 ; -
      01C5 512
      01C5 513 ERROR_CALL: ; Called by I format
003C 01C5 514 .WORD ^M<R2,R3,R4,R5>
      01C7 515
      01C7 516 ERROR:
61 50 2A 50 08 BC 7D 01C7 517 MOVQ @out_string(AP), R0 ; Get string descriptor
50 50 2A 6E 00 2C 01CB 518 MOVCS #0, (SP), #^A/*/, R0, (R1) ; Fill with *
      D0 01D1 519 MOVL #OTSS_OUTCONERR, R0 ; Output conversion error
      04 01D8 520 RET

```



```

01D9 522 .SBTTL OTSS$CVT_L_TI - Long to text, I format
01D9 523 :++
01D9 524 : FUNCTIONAL DESCRIPTION:
01D9 525 :
01D9 526 : This routine converts its input value to a text representation,
01D9 527 : using base 10 (decimal).
01D9 528 :
01D9 529 : OTSS$CVT_L_TI supports FORTRAN Iw and Iw.m output conversion.
01D9 530 :
01D9 531 : A separate entry point FOR$CNV_OUT_I is provided for compatibility
01D9 532 : with previous releases. Note that the input value for
01D9 533 : OTSS$CVT_L_TI is by reference while that for FOR$CNV_OUT_I is
01D9 534 : by value.
01D9 535 :
01D9 536 : CALLING SEQUENCE:
01D9 537 :
01D9 538 : status.wlc.v = OTSS$CVT_L_TI (value.rx.r, out_string.wt.ds
01D9 539 :                               [, int_digits.rl.v
01D9 540 :                               [, value_size.rl.v
01D9 541 :                               [, caller_flags.rbu.v]]])
01D9 542 :
01D9 543 : status.wlc.v = FOR$CNV_OUT_I (value.rl.v, out_string.wt.ds)
01D9 544 :
01D9 545 :
01D9 546 : INPUT PARAMETERS:
01D9 547 :
00000004 01D9 548 : value = 4
0000000C 01D9 549 : int_digits = 12
01D9 550 :
01D9 551 : ; Input value to be converted to text
01D9 552 : ; Minimum number of digits to be produced.
01D9 553 : ; If actual number of significant digits
01D9 554 : ; is smaller, leading zeroes will be
01D9 555 : ; produced. If int_digits is zero
01D9 556 : ; and value is zero, a blank field will
01D9 557 : ; result. The default is 1.
00000010 01D9 558 : value_size = 16
01D9 559 : ; The size of value in bytes. If
01D9 560 : ; present, value_size must be either
01D9 561 : ; 1, 2 or 4. If value_size is 1 or 2, the
01D9 562 : ; value is sign extended to a longword
01D9 563 : ; before conversion. The default is
01D9 564 : ; 4 if this argument is not present.
00000014 01D9 565 : caller_flags = 20
00000000 01D9 566 : V_FORCEPLUS = 0
01D9 567 : ; Flags supplied by caller:
01D9 568 : ; If set, a plus sign will be forced
01D9 569 : ; for positive values.
01D9 570 :
01D9 571 : IMPLICIT INPUTS:
01D9 572 :
01D9 573 : NONE
01D9 574 :
01D9 575 : OUTPUT PARAMETERS:
01D9 576 :
00000008 01D9 577 : out_string = 8
01D9 578 : ; Output string by descriptor.
01D9 579 :
01D9 580 : IMPLICIT OUTPUTS:
01D9 581 :
01D9 582 : NONE
01D9 583 :
01D9 584 : COMPLETION CODES:
01D9 585 :

```



```

01D9 579 : SSS NORMAL - Successful completion.
01D9 580 : OTSS_OUTCONERR - Output conversion error. Either the converted
01D9 581 : value did not fit in the field provided or the
01D9 582 : byte count was not 1, 2 or 4. The field is filled
01D9 583 : with asterisks.
01D9 584 :
01D9 585 : SIDE EFFECTS:
01D9 586 :
01D9 587 : NONE
01D9 588 :
01D9 589 : --
01D9 590 :
01D9 591 : +
01D9 592 : Definition of flag bits.
01D9 593 : -
01D9 594 :
00000008 01D9 595 V_NEGATIVE = 8 ; Bit to set in R5 if negative
00000100 01D9 596 M_NEGATIVE = 1aV_NEGATIVE ; Mask for V_NEGATIVE
01D9 597
007C 01D9 598 .ENTRY FOR$CNV_OUT_I, ^M<R2,R3,R4,R5,R6>
01DB 599
54 01 D0 01DB 600 MOVL #1, R4 ; Number of integer digits
55 D4 01DE 601 CLRL R5 ; No flags
50 04 AC D0 01E0 602 MOVL value(AP), R0 ; Value is in argument list
47 11 01E4 603 BRB COMMON_I ; Go to common routine
01E6 604
007C 01E6 605 .ENTRY OTSS$CVT_L_TI, ^M<R2,R3,R4,R5,R6>
01E8 606
03 02 55 D4 01E8 607 CLRL R5 ; Assume no flags
6C 8F 01EA 608 CASEB (AP), #2, #3 ; Select on argument count
000A' 01EE 609 1$: .WORD 20$-1$ ; 2 arguments
0035' 01F0 610 .WORD 44$-1$ ; 3 arguments
0017' 01F2 611 .WORD 40$-1$ ; 4 arguments
0013' 01F4 612 .WORD 50$-1$ ; 5 arguments
09 11 01F6 613 BRB 50$ ; assume >5 arguments
01F8 614 20$:
54 01 D0 01F8 615 MOVL #1, R4 ; Get integer digits
50 04 BC D0 01FB 616 MOVL @value(AP), R0 ; longword value
2C 11 01FF 617 BRB COMMON_I
55 14 AC 90 0201 618 50$: MOVB caller_flags(AP), R5 ; Get flags
04 00 10 AC CF 0205 619 40$: CASEL value_size(AP), #0, #4 ; Select on value size
0019' 020A 620 2$: .WORD 44$-2$ ; 0 - assume 4 bytes
000D' 020C 621 .WORD 41$-2$ ; 1 byte
0013' 020E 622 .WORD 42$-2$ ; 2 bytes
007D' 0210 623 .WORD ERROR I-2$ ; 3 bytes, error
0019' 0212 624 .WORD 44$-2$ ; 4 bytes
50 0070 31 0214 625 BRW ERROR I ; other, error
50 04 BC 98 0217 626 41$: CVTBL @value(AP), R0 ; Convert byte
0A 11 021B 627 BRB 30$ ; Continue
50 04 BC 32 021D 628 42$: CVTWL @value(AP), R0 ; Convert word
04 11 0221 629 BRB 30$ ; Continue
50 04 BC D0 0223 630 44$: MOVL @value(AP), R0 ; Convert longword
54 0C AC D0 0227 631 30$: MOVL int_digits(AP), R4 ; Get integer digits
50 D5 022B 632 TSTL R0 ; Set condition codes for test
022D 633
022D 634 COMMON_I:
08 18 022D 635 BGEQ COMMON_IU ; Skip if value positive

```



```

55 0100 8F A8 022F 636 BSW2 #M_NEGATIVE, R5 ; Indicate negative
50 50 CE 0234 637 MNEGL R0, R0 ; Get absolute value
;+
51 D4 0237 638 COMMON_IU: ; Clear low-order part of value
52 08 BC 7D 0239 639 CLRL R1 ; Get descriptor in R2-R3
52 52 3C 023D 640 MOVQ @out_string(AP), R2 ; Get length in R2
53 52 C0 0240 641 MOVZWL R2, R2 ; Get address of last+1 character
53 52 C0 0240 642 ADDL2 R2, R3 ; Store digits
53 52 OF 11 0243 643 BRB 15$
0245 644 ;+
0245 645 ;+ Store all significant digits.
0245 646 ;+
0245 647 ;+
56 50 50 0A 7B 0245 648 10$: EDIV #10, R0, R0, R6 ; Get quotient in R0-R1, remainder in R6
52 D7 024A 649 DECL R2 ; Decrement length
39 19 024C 650 BLSS ERROR_I ; Error if no more chars left
73 56 30 81 024E 651 ADDB3 #^A/07, R6, -(R3) ; Store next digit
54 D7 0252 652 DECL R4 ; Decrement zero-fill count
50 D5 0254 653 15$: TSTL R0 ; Are we done now?
ED 12 0256 654 BNEQ 10$ ; Loop if not
07 11 0258 655 BRB 25$ ; Fill in leading zeroes
025A 656 ;+
025A 657 ;+ Fill in any leading zeroes needed. R4 has remaining zero count.
025A 658 ;+
025A 659 ;+
52 D7 025A 660 20$: DECL R2 ; 1 less character
29 19 025C 661 BLSS ERROR_I ; Have we run out?
73 30 90 025E 662 MOVB #^A/07, -(R3) ; Move a zero
F6 54 F4 0261 663 25$: SOBGEQ R4, 20$ ; Loop till done
0264 664 ;+
0264 665 ;+ Store sign, if needed.
0264 666 ;+
0264 667 ;+
09 55 08 E0 0264 668 BBS #V_NEGATIVE, R5, 30$ ; Skip if value negative
14 55 00 E1 0268 669 BBC #V_FORCEPLUS, R5, 50$ ; Test for forced plus
50 2B 90 026C 670 MOVB #^A/+/ , R0 ; Use plus sign
03 11 026F 671 BRB 35$ ; Rejoin common code
50 2D 90 0271 672 30$: MOVB #^A/-/ , R0 ; Use minus sign
52 D7 0274 673 35$: DECL R2 ; 1 less character
OF 19 0276 674 BLSS ERROR_I ; Have we run out?
73 50 90 0278 675 MOVB R0, -(R3) ; Move sign, '-' or '+'
03 11 027B 676 BRB 50$ ; Blank fill, if needed.
027D 677 ;+
027D 678 ;+ Blank fill remainder, if needed. R2 has remaining blank count.
027D 679 ;+
027D 680 ;+
73 20 90 027D 681 40$: MOVB #^A/ / , -(R3) ; Move a blank
FA 52 F4 0280 682 50$: SOBGEQ R2, 40$ ; Loop till done
50 01 D0 0283 683 MOVL #1, R0 ; Success
04 04 0286 684 RET
0287 685
FF39 CF 6C FA 0287 686 ERROR_I: CALLG (AP), W^ERROR_CALL ; Fill with asterisks
04 04 028C 687 RET ; with error status in R0
028C 688
```



```

028D 690 .SBTTL OTSS$CVT_L_TU - Long to text, unsigned decimal format
028D 691 :++
028D 692 : FUNCTIONAL DESCRIPTION:
028D 693 :
028D 694 : This routine converts its input value to a text representation,
028D 695 : using unsigned base 10 (decimal).
028D 696 :
028D 697 : CALLING SEQUENCE:
028D 698 :
028D 699 : status.wlc.v = OTSS$CVT_L_TU (value.rx.r, out_string.wt.ds
028D 700 : [, int_digits.rl.v
028D 701 : [, value_size.rl.v
028D 702 : [, caller_flags.rbu.v]]])
028D 703 :
028D 704 :
028D 705 : INPUT PARAMETERS:
028D 706 :
00000004 028D 707 : value = 4 ; Input value to be converted to text
0000000C 028D 708 : int_digits = 12 ; Minimum number of digits to be produced.
028D 709 : ; If actual number of significant digits
028D 710 : ; is smaller, leading zeroes will be
028D 711 : ; produced. If int_digits is zero
028D 712 : ; and value is zero, a blank field will
028D 713 : ; result. The default is 1.
00000010 028D 714 : value_size = 16 ; The size of value in bytes. If
028D 715 : ; present, value_size must be either
028D 716 : ; 1, 2 or 4. If value_size is 1 or 2, the
028D 717 : ; value is sign extended to a longword
028D 718 : ; before conversion. The default is
00000014 028D 719 : caller_flags = 20 ; 4 if this argument is not present.
028D 720 : ; Flags supplied by caller:
028D 721 :
028D 722 :
028D 723 : IMPLICIT INPUTS:
028D 724 :
028D 725 : NONE
028D 726 :
028D 727 : OUTPUT PARAMETERS:
028D 728 :
00000008 028D 729 : out_string = 8 ; Output string by descriptor.
028D 730 :
028D 731 : IMPLICIT OUTPUTS:
028D 732 :
028D 733 : NONE
028D 734 :
028D 735 : COMPLETION CODES:
028D 736 :
028D 737 : SSS$ NORMAL - Successful completion.
028D 738 : OTSS$_OUTCONERR - Output conversion error. Either the converted
028D 739 : ; value did not fit in the field provided or the
028D 740 : ; byte count was not 1, 2 or 4. The field is filled
028D 741 : ; with asterisks.
028D 742 :
028D 743 : SIDE EFFECTS:
028D 744 :
028D 745 : NONE
028D 746 :

```



```

028D 747 :--
028D 748 :--
028D 749 :+
028D 750 : Definition of flag bits.
028D 751 :-
028D 752 :
028D 753 :
007C 028D 754 .ENTRY OTSS$CVT_L_TU, ^M<R2,R3,R4,R5,R6>
028F 755 :
03 02 55 D4 028F 756 CLRL R5 : Assume no flags
6C 8F 0291 757 CASEB (AP), #2, #3 : Select on argument count
000A' 0295 758 1$: .WORD 20$-1$ : 2 arguments
0032' 0297 759 .WORD 44$-1$ : 3 arguments
0014' 0299 760 .WORD 40$-1$ : 4 arguments
0014' 029B 761 .WORD 40$-1$ : 5 arguments
0A 11 029D 762 BRB 40$ : assume >5 arguments
029F 763 20$:
50 54 01 D0 029F 764 MOVL #1, R4 : Get integer digits
04 04 BC D0 02A2 765 MOVL @value(AP), R0 : longword value
FF8E 31 02A6 766 BRW COMMON_IU
04 00 10 AC CF 02A9 767 40$: CASEL value_size(AP), #0, #4 : Select on value size
0019' 02AE 768 2$: .WORD 44$-2$ : 0 - assume 4 bytes
000D' 02B0 769 .WORD 41$-2$ : 1 byte
0013' 02B2 770 .WORD 42$-2$ : 2 bytes
FFD9 02B4 771 .WORD ERROR_I-2$ : 3 bytes, error
0019' 02B6 772 .WORD 44$-2$ : 4 bytes
FFCC 31 02B8 773 BRW ERROR_I : other, error
50 04 BC 9A 02BB 774 41$: MOVZBL @value(AP), R0 : Convert byte
0A 11 02BF 775 BRB 30$ : Continue
50 04 BC 3C 02C1 776 42$: MOVZWL @value(AP), R0 : Convert word
04 11 02C5 777 BRB 30$ : Continue
50 04 BC D0 02C7 778 44$: MOVL @value(AP), R0 : Convert longword
54 0C AC D0 02CB 779 30$: MOVL int_digits(AP), R4 : Get integer digits
FF65 31 02CF 780 BRW COMMON_IU : Join common code

```



```
02D2 782 .SBTTL OTSS$CVT_L_TL - Long to text, L format
02D2 783 :++
02D2 784 : FUNCTIONAL DESCRIPTION:
02D2 785 :
02D2 786 : This routine converts its input value to a text representation,
02D2 787 : using FORTRAN L (logical) format.
02D2 788 :
02D2 789 : The output field will consist of (width-1) blanks followed by:
02D2 790 : the letter T if the lowest bit is set;
02D2 791 : the letter F if the lowest bit is clear.
02D2 792 :
02D2 793 : A separate entry point FOR$CNV_OUT_L is provided for compatibility
02D2 794 : with previous releases. Note that the input value for
02D2 795 : OTSS$CVT_L_TL is by reference while that for FOR$CNV_OUT_L is
02D2 796 : by value.
02D2 797 :
02D2 798 : CALLING SEQUENCE:
02D2 799 :
02D2 800 : status.wlc.v = OTSS$CVT_L_TL (value.rL.r, out_string.wt.ds)
02D2 801 :
02D2 802 : status.wlc.v = FOR$CNV_OUT_L (value.rl.v, out_string.wt.ds)
02D2 803 :
02D2 804 :
02D2 805 : INPUT PARAMETERS:
02D2 806 :
00000004 02D2 807 : value = 4 ; Input value to be converted to text
02D2 808 :
02D2 809 : IMPLICIT INPUTS:
02D2 810 :
02D2 811 : NONE
02D2 812 :
02D2 813 : OUTPUT PARAMETERS:
02D2 814 :
00000008 02D2 815 : out_string = 8 ; Output string by descriptor.
02D2 816 :
02D2 817 : IMPLICIT OUTPUTS:
02D2 818 :
02D2 819 : NONE
02D2 820 :
02D2 821 : COMPLETION CODES:
02D2 822 :
02D2 823 : SS$ NORMAL - Successful completion.
02D2 824 : OTSS$_OUTCONERR - Output conversion error. This can only occur
02D2 825 : if the output string is of zero length.
02D2 826 :
02D2 827 : SIDE EFFECTS:
02D2 828 :
02D2 829 : NONE
02D2 830 :
02D2 831 : --
02D2 832 :
0004 02D2 833 : .ENTRY FOR$CNV_OUT_L, ^M<R2>
02D4 834 :
52 04 AC 90 02D4 835 : MOVB value(AP), R2 ; Get low byte of value
06 11 02D8 836 : BRB COMMON_L ; Go to common routine
02DA 837 :
0004 02DA 838 : .ENTRY OTSS$CVT_L_TL , ^M<R2>
```



```

52 04 BC 90 02DC 839      MOVB    avalue(AP), R2      ; Get low byte of value
              02DC 840
              02E0 841
              02E0 842
              02E0 843 COMMON_L:
50 08 BC 7D 02E0 844      MOVQ    @out_string(AP), R0    ; Get descriptor
50 50 3C 02E4 845      MOVZWL   R0, R0                  ; Length
              1E 15 02E7 846      BLEQ    ERROR_L        ; Zero length, error
51 06 52 C0 02E9 847      ADDL2    R0, RT                ; 1 byte past end of string
71 46 8F 90 02EC 848      BLBS     R2, TRUE              ; TRUE or FALSE?
              04 11 02F3 849      MOVB    #^A/F/, -(R1)   ; result is F
71 54 8F 90 02F5 851 TRUE: MOVB    #^A/T/, -(R1)         ; result is T
              02F9 852
              02F9 853 EXIT_L:
              02F9 854      DECL     R0                  ; 1 less character
              02FB 855      BEQL     20$                  ; All done?
              71 20 90 02FD 856 10$: MOVB    #^A/ /, -(R1) ; Blank fill
              FA 50 F5 0300 857      SOBGTR  R0, 10$       ; Loop till done
50 50 01 D0 0303 858 20$: MOVL     #1, R0                ; Success
              04 04 0306 859      RET                     ; Return
              0307 860
              0307 861 ERROR_L:
50 00000000'8F D0 0307 862      MOVL     #OTSS_OUTCONERR, R0 ; Output conversion error
              04 04 030E 863      RET                     ; Return
              030F 864
              030F 865
              030F 866      .END

```

OTSSCVTLT
Symbol table

- Convert longword to text, 0, Z, L, B, M 16 16-SEP-1984 00:24:59 VAX/VMS Macro V04-00 Page 21
6-SEP-1984 11:13:06 [LIBRTL.SRC]OTSCVTLT.MAR;1 (10)

```

BIT OFFSET      = 00000018
COMMON_I        0000022D R    01
COMMON_IU       00000237 R    01
COMMON_L        000002E0 R    01
COMMON_O        00000027 R    01
COMMON_Z        000000AB R    01
ERROR           000001C7 R    01
ERROR_CALL      000001C5 R    01
ERROR_I         00000287 R    01
ERROR_L         00000307 R    01
EXIT_B          00000159 R    01
EXIT_L          000002F9 R    01
EXIT_O          00000079 R    01
EXIT_Z          00000118 R    01
FLAGS           = 00000014
FOR$CNV_OUT_I   000001D9 RG   01
FOR$CNV_OUT_L   000002D2 RG   01
FOR$CNV_OUT_O   00000010 RG   01
FOR$CNV_OUT_Z   00000094 RG   01
INITIALIZE      00000174 R    01
INT DIGITS      = 0000000C
LETTERS         00000000 R    01
M NEGATIVE      = 00000100
OTSSCVT_L_TB    00000133 RG   01
OTSSCVT_L_TI    000001E6 RG   01
OTSSCVT_L_TL    000002DA RG   01
OTSSCVT_L_TO    0000001E RG   01
OTSSCVT_L_TU    0000028D RG   01
OTSSCVT_L_TZ    000000A2 RG   01
OTSS OUTCONERR ***** X    00
OUT_STRING      = 00000008
TRUE            000002F5 R    01
VALUE           = 00000004
VALUE_SIZE      = 00000010
V_FORCEPLUS     = 00000000
V_NEGATIVE      = 00000008
V_SIZE_IN_BITS  = 00000002
ZERO_FILL      000001B4 R    01
  
```

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

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPI	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
_OTSSCODE	0000030F (783.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG				

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:02.11
Command processing	106	00:00:00.31	00:00:02.72
Pass 1	100	00:00:01.23	00:00:03.77

Symbol table sort	0	00:00:00.05	00:00:00.26
Pass 2	154	00:00:01.00	00:00:03.60
Symbol table output	4	00:00:00.03	00:00:00.65
Psect synopsis output	3	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	398	00:00:02.69	00:00:13.13

The working set limit was 1200 pages.

13930 bytes (28 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 39 non-local and 58 local symbols.

866 source lines were read in Pass 1, producing 38 object records in Pass 2.

0 pages of virtual memory were used to define 0 macros.

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

Macro library name

Macros defined

_\$255\$DUA28:[SYSLIB]STARLET.MLB;2

0

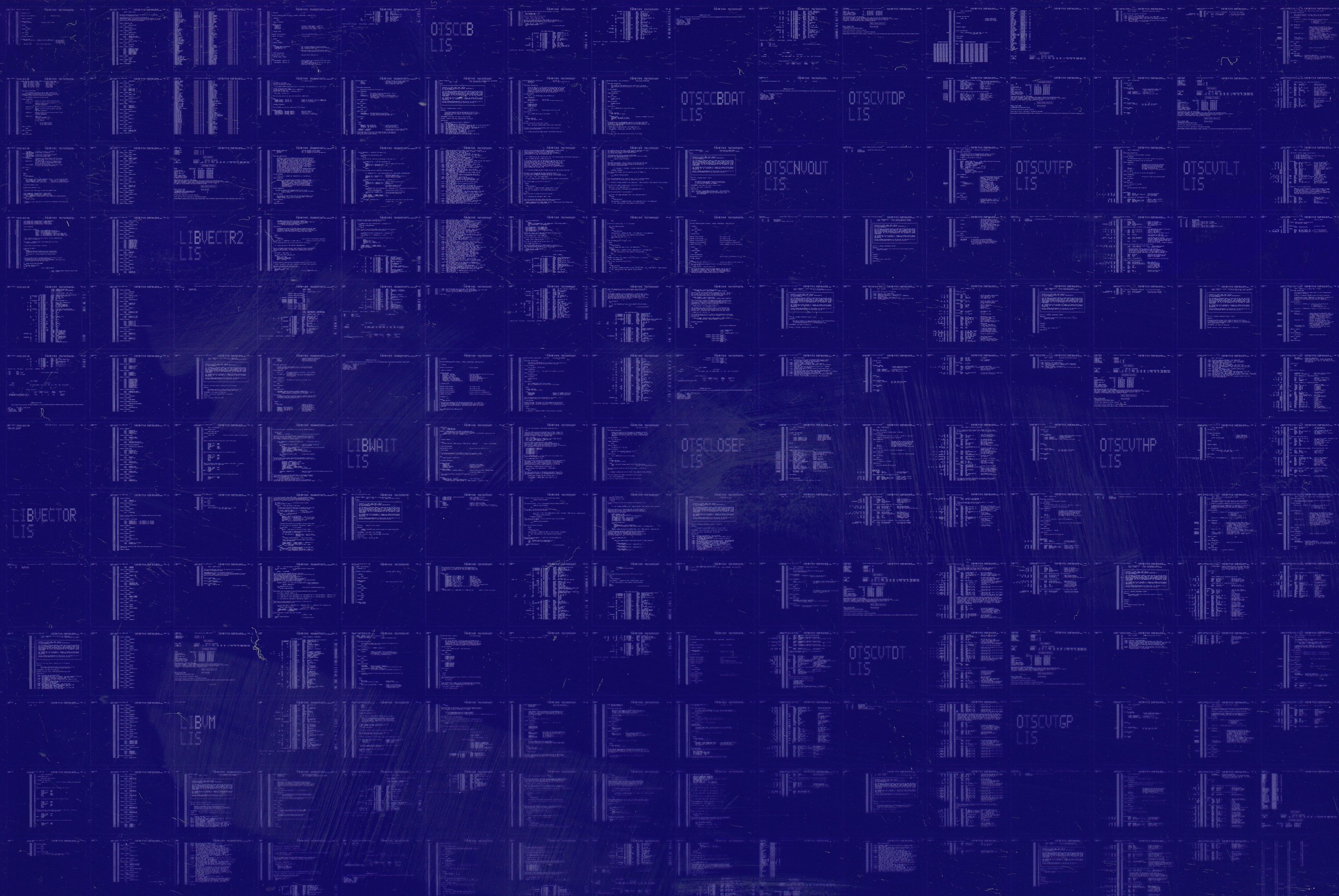
0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSCVTLT/OBJ=OBJ\$:OTSCVTLT MSRC\$:OTSCVTLT/UPDATE=(ENH\$:OTSCVTLT)

0211 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0212 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

