


```

BBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFFFF      000000      RRRRRRRR      IIIIII      NN      NN      TTTTTTTTTT
BBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFFFF      000000      RRRRRRRR      IIIIII      NN      NN      TTTTTTTTTT
BB      BB      AA      AA      SS      FF      00      00      RR      RR      II      NN      NN      TT
BB      BB      AA      AA      SS      FF      00      00      RR      RR      II      NN      NN      TT
BB      BB      AA      AA      SS      FF      00      00      RR      RR      II      NNNN      NN      TT
BBBBBBBB      AA      AA      SSSSSS      FFFFFFFF      00      00      RRRRRRRR      II      NN      NN      TT
BBBBBBBB      AA      AA      SSSSSS      FFFFFFFF      00      00      RRRRRRRR      II      NN      NN      TT
BB      BB      AAAAAAAAAA      SS      FF      00      00      RR      RR      II      NN      NNNN      TT
BB      BB      AAAAAAAAAA      SS      FF      00      00      RR      RR      II      NN      NNNN      TT
BB      BB      AA      AA      SS      FF      00      00      RR      RR      II      NN      NN      TT
BB      BB      AA      AA      SS      FF      00      00      RR      RR      II      NN      NN      TT
BBBBBBBB      AA      AA      SSSSSSSS      FF      00      00      RR      RR      IIIIII      NN      NN      TT
BBBBBBBB      AA      AA      SSSSSSSS      FF      000000      RR      RR      IIIIII      NN      NN      TT

```

```

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

```

```

....
....
....
....

```

```

1 0001 0 %TITLE 'BAS$$FORMAT_INT - Basic format interpreter'
2 0002 0 MODULE BAS$$FOR_INT (
3 0003 0 ! Basic format interpreter
4 0004 0 ! IDENT = '2-013' ! File: BASFORINT.B32 EDIT:MDL2013
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY:
33 0033 1
34 0034 1 Basic support library - not user callable
35 0035 1
36 0036 1 ABSTRACT:
37 0037 1
38 0038 1 This is the format interpreter for Basic PRINT USING and FORMATS.
39 0039 1
40 0040 1 ENVIRONMENT: User access mode, AST reentrant
41 0041 1
42 0042 1 AUTHOR: Donald G. Petersen, CREATION DATE: 03-May-79
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 DGP,03-May-79 : VERSION 1
47 0047 1 1-001 - original
48 0048 1 1-002 - Get TPAMAC from SYSS$LIBRARY. JBS 22-MAY-1979
49 0049 1 1-003 - Add PSECT declaration and make BAS$$FORMAT_INT global. DGP 22-May-79
50 0050 1 1-004 - Remove comment from switches statement. DGP 22-May-79
51 0051 1 1-005 - The TPARSE parameter block is passed by value using a CALLG. Change
52 0052 1 the MAP of the formal in each action routine. DGP 23-May-79
53 0053 1 1-006 - The TPARSE argument list for the action routines is not quite right.
54 0054 1 AP is needed as a BUILTIN. DGP 29-May-79
55 0055 1 1-007 - Change error routine call to BAS$$STOP from BAS$$STOP IO 'cuz
56 0056 1 the format interpreter knows nothing about I/O. DGP 29-May-79
57 0057 1 1-008 - Add BAS$$ prefix to PUKEYWDS and PRINT_USING. DGP 30-May-79

```

```
58 0058 1 1-009 - Change the names of the STR linkages. JBS 04-JUN-1979
59 0059 1 1-010 - Change length of K_INTER_STO_LEN to be 28. Bug fix. DGP 12-Jul-79
60 0060 1 1-011 - Make PERCENT be a macro rather than a BIND. JBS 14-JUL-1979
61 0061 1 1-012 - Change calls to STR$COPY. JBS 16-JUL-1979
62 0062 1 1-013 - Make PERCENT a BIND rather than a macro. DGP 16-Jul-79
63 0063 1 1-014 - Work on numerics some. DGP 16-Jul-79
64 0064 1 1-015 - Add V_PERIOD. DGP 20-Jul-79
65 0065 1 1-016 - Bug fix in double precision floating. DGP 30-Jul-79
66 0066 1 1-017 - Yes, E format can have a (negative) number which won't fit. DGP
67 0067 1 30-Jul-79
68 0068 1 1-018 - Retro fit floating into integer. DGP 01-Aug-79
69 0069 1 1-019 - Use REQUIRE rather than LIBRARY to read TPAMAC, so we can try
70 0070 1 new BLISS compiler. JBS 06-SEP-1979
71 0071 1 1-020 - Go back to using LIBRARY, but use a logical name. JBS 12-SEP-1979
72 0072 1 1-021 - Initialize all null string desc. pointers to 0. DGP 18-Sep-79
73 0073 1 1-022 - Translate SYSS$CURRENCY, SYSS$RADIX_POINT, and SYSS$DIGIT_SEP. DGP
74 0074 1 30-Oct-79
75 0075 1 1-023 - Return all temporary strings allocated from HEAP store. DGP 08-Nov-79
76 0076 1 1-024 - Receive the scale factor and pass it along to the conversion routines.
77 0077 1 DGP 25-Nov-79
78 0078 1 2-001 - Add routines to use SPANC, SCANC, etc. instructions instead of
79 0079 1 LIB$TPARSE for performance optimizations. Replace use of STR$CONCAT
80 0080 1 with STR$APPEND in OUTPUT_ARG where possible. DJB 08-Jun-1981.
81 0081 1 2-002 - Add support for byte, g floating, and h floating. PLL 26-Aug-81
82 0082 1 2-003 - Add f format for h floating. PLL 8-Sep-81
83 0083 1 2-004 - Add <CD>, <%>, <0>, and _ format characters. PLL 23-Nov-81
84 0084 1 2-005 - Add support for packed decimal. PLL 12-Jan-82
85 0085 1 2-006 - Change calls to STR$APPEND to STR$CONCAT so that static string destinations
86 0086 1 will work. PLL 20-Apr-82
87 0087 1 2-007 - allow both <CD> and <cd> format. MDL 9-Aug-1982
88 0088 1 2-008 - fix bug in processing underscores; shouldn't EXITLOOP in SPAN_CONSTANT
89 0089 1 after handling underscore as there may be more constant following it.
90 0090 1 MDL 10-Aug-1982
91 0091 1 2-009 - Fix section in SPAN_CONSTANT on underscores.
92 0092 1 A new class was added to CLASS_TABLE, mask special, so that SCANC
93 0093 1 will pick up any format character and not just characters which
94 0094 1 begin a format sequence, for underscore. PLL 28-Sep-1982
95 0095 1 2-010 - SPAN_BRACKET should only look ahead 1 character when searching
96 0096 1 for the start of a valid <> sequence. Other formatting characters
97 0097 1 may be bypassed in the case where <> is in an unexpected place.
98 0098 1 SPAN_CONSTANT must check for multi-char format sequences after an
99 0099 1 underscore. For instance, '$$' must appear in multiples - a single
100 0100 1 '$' is not a valid format sequence. So if '_$$' appears, include
101 0101 1 both '$$' in the constant.
102 0102 1 Yet another class was added to CLASS_TABLE, so that characters
103 0103 1 which follow '<' can be scanned.
104 0104 1 PLL 5-Oct-1982
105 0105 1 2-011 - make <CD> at the beginning of a format sequence a constant in
106 0106 1 order to be consistent with BP2, rather than an error.
107 0107 1 MDL 9-Dec-1982
108 0108 1 2-012 - fix case of $$<0>#. BP2 treats this as 2 fields; the first being
109 0109 1 a dollar sign followed by a single digit, and the second a 3-digit
110 0110 1 field with leading zeroes. Previously, we had treated this as a
111 0111 1 single field. MDL 10-Dec-1982
112 0112 1 2-013 - Fix case of a single underscore not followed by anything. We
113 0113 1 should just ignore this. Also fix case of <CD> at the end of
114 0114 1 a format string; both these cases cause infinite loops.
```

```
: 115      0115 1 !      It is now apparent that whereas VAX-11 BASIC Run-Time code treats
: 116      0116 1 !      underscores not followed by format characters as literals, BP2
: 117      0117 1 !      treats these as something to be ignored. MDL 18-Mar-1983
: 118      0118 1 ! --
: 119      0119 1 !
: 120      0120 1 !<BLF/PAGE>
```

```

: 122 0121 1 %SBTTL 'Declarations'
: 123 0122 1
: 124 0123 1  SWITCHES:
: 125 0124 1
: 126 0125 1
: 127 0126 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
: 128 0127 1
: 129 0128 1
: 130 0129 1  LINKAGES:
: 131 0130 1
: 132 0131 1
: 133 0132 1 REQUIRE 'RTLIN:OTSLNK';           ! macros for all linkages except STR$
: 134 0561 1
: 135 0562 1 REQUIRE 'RTLIN:STRLNK';       ! macros for STR$ linkages
: 136 0747 1
: 137 0748 1
: 138 0749 1  TABLE OF CONTENTS:
: 139 0750 1
: 140 0751 1
: 141 0752 1 FORWARD ROUTINE
: 142 0753 1   BAS$$FORMAT_INT : NOVALUE,       ! the Basic format interpreter
: 143 0754 1
: 144 0755 1  +
: 145 0756 1  | A possible optimization in the future could be to make these internal
: 146 0757 1  | routines JSB entry points. This should be done after the full range
: 147 0758 1  | of planned format characters is implemented (or allowed for).
: 148 0759 1  | -
: 149 0760 1
: 150 0761 1   DECODE_PROCESS_FORMAT,           ! main driving routine
: 151 0762 1   SPAN_CONSTANT,                 ! find start of formatting sequence
: 152 0763 1   SPAN_SINGLE_QUOTE,            ! process single quote format classes
: 153 0764 1   SPAN_TEXT,                    ! span various text format classes
: 154 0765 1   SPAN_BACKSLASH,              ! process backslash format class
: 155 0766 1   SPAN_BRACKET,                ! process < > format classes
: 156 0767 1   SPAN_NUMERIC,                ! process numeric format class
: 157 0768 1   OUTPUT_ARG;                  ! do all of the formatting
: 158 0769 1
: 159 0770 1
: 160 0771 1  INCLUDE FILES:
: 161 0772 1
: 162 0773 1
: 163 0774 1 REQUIRE 'RTLIN:RTPSECT';       ! Declare Psects macros
: 164 0869 1
: 165 0870 1 LIBRARY 'RTLSTARLE';          ! STARLET library for macros and symbols
: 166 0871 1
: 167 0872 1
: 168 0873 1  MACROS:
: 169 0874 1
: 170 0875 1
: 171 0876 1  MACRO
: 172 0877 1   RET_STR = 0, 0, 32, 0%,
: 173 0878 1   PU_MASK = 4, 0, 32, 0%,
: 174 0879 1   ELEM_TYPE = 8, 0, 32, 0%,
: 175 0880 1   ELEM = 12, 0, 32, 0%,
: 176 0881 1   FRACTION_DIGIT = 16, 0, 32, 1%,
: 177 0882 1   INTEGER_DIGITS = 20, 0, 32, 1%,
: 178 0883 1   CHARACTER = 24, 0, 32, 1%,

```

```

: 179 0884 1 SCALE_FACTOR = 28, 0, 32, 1%,
: 180 0885 1 NEXT_CHAR = 32, 0, 32, 0%;
: 181 0886 1
: 182 0887 1
: 183 0888 1 EQUATED SYMBOLS:
: 184 0889 1
: 185 0890 1
: 186 0891 1 LITERAL
: 187 0892 1
: 188 0893 1
: 189 0894 1 +
: 190 0895 1 These literals define the bits in the PRINT USING mask (PU_MASK)
: 191 0896 1 NOTE: These are well ordered. Action routines take advantage of the
: 192 0897 1 fact that:
: 193 0898 1 F format is < E_MASK
: 194 0899 1 E_MASK <= E format < R JUSTIFY_MASK
: 195 0900 1 Text is >= R JUSTIFY_MASK
: 196 0901 1 Be careful when adding new values
: 197 0902 1
: 198 0903 1 NUMERIC_MASK = 1, numeric format
: 199 0904 1 MINUS_MASK = 2, trailing minus required
: 200 0905 1 COMMA_MASK = 4, digit group separator required
: 201 0906 1 DOLLAR_MASK = 8, floating currency required
: 202 0907 1 STAR_MASK = 16, asterisk fill required
: 203 0908 1
: 204 0909 1
: 205 0910 1 +
: 206 0911 1 bit 5 (value = 32) should not be used unless we run out of bits
: 207 0912 1 The conversion routines use almost the same bits for the same things
: 208 0913 1 and bit 5 means strip_trailing_spaces.
: 209 0914 1
: 210 0915 1 PERIOD_MASK = 64, a period is in the format
: 211 0916 1 E_MASK = 128, number should be in E format
: 212 0917 1 R_JUSTIFY_MASK = 256, right justify the string
: 213 0918 1 C_JUSTIFY_MASK = 512, center justify the string
: 214 0919 1 EXTEND_MASK = 1024, extend the string as necessary
: 215 0920 1 L_JUSTIFY_MASK = 2048, left justify the string
: 216 0921 1 PERCENT_MASK = 4096, suppress zeroes mask
: 217 0922 1 ZERO_MASK = 8192, print leading zeroes mask
: 218 0923 1 CD_MASK = 16384, credit/debit mask
: 219 0924 1 K_CVT_FLAGS = 28767, turn off flags not interesting to
: 220 0925 1 conversion routine
: 221 0926 1 K_INTER_STO_LEN = 36, length of local interpreter store
: 222 0927 1 in bytes
: 223 0928 1 K_AST_FLAG = 1, 0 - do not disable ASTs
: 224 0929 1 1 - disable ASTs
: 225 0930 1
: 226 0931 1 +
: 227 0932 1 ASCII character codes.
: 228 0933 1
: 229 0934 1
: 230 0935 1 CHAR_SPACE = 32 : UNSIGNED (8),
: 231 0936 1 CHAR_EXCLAMATION = 33 : UNSIGNED (8),
: 232 0937 1 CHAR_POUND = 35 : UNSIGNED (8),
: 233 0938 1 CHAR_DOLLAR = 36 : UNSIGNED (8),
: 234 0939 1 CHAR_SINGLE_QUOTE = 39 : UNSIGNED (8),
: 235 0940 1 CHAR_STAR = 42 : UNSIGNED (8),

```

```

: 236 0941 1 CHAR_COMMA = 44 : UNSIGNED (8),
: 237 0942 1 CHAR_MINUS = 45 : UNSIGNED (8),
: 238 0943 1 CHAR_PERIOD = 46 : UNSIGNED (8),
: 239 0944 1 CHAR_BACKSLASH = 92 : UNSIGNED (8),
: 240 0945 1 CHAR_CARAT = 94 : UNSIGNED (8),
: 241 0946 1 CHAR_C = 67 : UNSIGNED (8),
: 242 0947 1 CHAR_E = 69 : UNSIGNED (8),
: 243 0948 1 CHAR_L = 76 : UNSIGNED (8),
: 244 0949 1 CHAR_R = 82 : UNSIGNED (8),
: 245 0950 1 CHAR_LOWER_C = 99 : UNSIGNED (8),
: 246 0951 1 CHAR_LOWER_E = 101 : UNSIGNED (8),
: 247 0952 1 CHAR_LOWER_L = 108 : UNSIGNED (8),
: 248 0953 1 CHAR_LOWER_R = 114 : UNSIGNED (8),
: 249 0954 1 CHAR_LF_ANGLE = 60 : UNSIGNED (8),
: 250 0955 1 CHAR_RT_ANGLE = 62 : UNSIGNED (8),
: 251 0956 1 CHAR_PERCENT = 37 : UNSIGNED (8),
: 252 0957 1 CHAR_ZERO = 48 : UNSIGNED (8),
: 253 0958 1 CHAR_UNDERSCORE = 95 : UNSIGNED (8),
: 254 0959 1 CHAR_D = 68 : UNSIGNED (8),
: 255 0960 1
: 256 0961 1 !+
: 257 0962 1 !: Functions for numeric format evaluation (see SPAN_NUMERIC routine).
: 258 0963 1 !-
: 259 0964 1
: 260 0965 1 K_SPAN_INTEGER = 1,
: 261 0966 1 K_SPAN_INTEGER_NO_E = 2,
: 262 0967 1 K_SPAN_FRACTION = 3,
: 263 0968 1 K_SPAN_FRACTION_NO_E = 4,
: 264 0969 1
: 265 0970 1 !+
: 266 0971 1 !: Classes for SPANC/SCANC table.
: 267 0972 1 !-
: 268 0973 1
: 269 0974 1 MASK_FORMAT_CHARS = 1 : UNSIGNED (8),
: 270 0975 1 MASK_L = 2 : UNSIGNED (8),
: 271 0976 1 MASK_C = 4 : UNSIGNED (8),
: 272 0977 1 MASK_R = 8 : UNSIGNED (8),
: 273 0978 1 MASK_E = 16 : UNSIGNED (8),
: 274 0979 1 MASK_SPECIAL = 32 : UNSIGNED (8),
: 275 0980 1 MASK_LF_ANGLE = 64 : UNSIGNED (8);
: 276 0981 1
: 277 0982 1 !:
: 278 0983 1 !: PSECT DECLARATIONS:
: 279 0984 1 !:
: 280 0985 1
: 281 0986 1 DECLARE_PSECTS (BAS);
: 282 0987 1
: 283 0988 1 !:
: 284 0989 1 !: OWN STORAGE:
: 285 0990 1 !:
: 286 0991 1
: 287 0992 1 OWN
: 288 0993 1 CURRENCY_DESC, ! descriptor for currency symbol
: 289 0994 1 CU D
: 290 0995 1 DIGIT_SEP_DESC, ! descriptor for digit group separator
: 291 0996 1 DI D
: 292 0997 1 RADIX_PT_DESC, ! descriptor for radix point

```

```

293 0998 1 RA_D;
294 0999 1
295 1000 1 BIND
296 1001 1 E_FORMAT = UPLIT BYTE (REP 4 OF (CHAR_CARAT)),
297 1002 1 PERCENT = UPLIT (BYTE ('%')),
298 1003 1
299 1004 1
300 1005 1 +
301 1006 1 Set up the class table for all groups of characters that will be involved
302 1007 1 in uses of the SCANC and SPANC instructions. This class table is built
303 1008 1 into a PLIT. An example of OWN storage initialization at run time is
304 1009 1 given below to help clarify the classes set up in the PLIT.
305 1010 1 INCR I FROM 0 TO 255 DO
306 1011 1 SELECTONEU .I OF
307 1012 1 SET
308 1013 1 [CHAR_DOLLAR, CHAR_POUND, CHAR_PERIOD, CHAR_SINGLE_QUOTE,
309 1014 1 CHAR_EXCLAMATION, CHAR_BACKSLASH, CHAR_STAR] :
310 1015 1 CLASS_TABLE [.I] = MASK_FORMAT_CHARS;
311 1016 1 [CHAR_L, CHAR_LOWER_L] : CLASS_TABLE [.I] = MASK_L;
312 1017 1 [CHAR_C, CHAR_LOWER_C] : CLASS_TABLE [.I] = MASK_C;
313 1018 1 [CHAR_R, CHAR_LOWER_R] : CLASS_TABLE [.I] = MASK_R;
314 1019 1 [CHAR_E, CHAR_LOWER_E] : CLASS_TABLE [.I] = MASK_E;
315 1020 1 [OTHERWISE] : CLASS_TABLE [.I] = 0;
316 1021 1 TES;
317 1022 1
318 1023 1 Currently 5 formatting classes are used leaving 3 available.
319 1024 1 If more than 8 formatting classes are needed, a second CLASS_TABLE
320 1025 1 will have to be used unless various formatting classes can be grouped
321 1026 1 together (e.g. if MASK_C + MASK_R is passed to the SPANC instruction
322 1027 1 as the mask, all of the four characters CHAR_C, CHAR_LOWER_C, CHAR_R,
323 1028 1 CHAR_LOWER_R would be spanned.
324 1029 1
325 1030 1 To add a new format character (or group of characters), the SPAN_CONSTANT
326 1031 1 routine should be modified to properly look for the beginning of the
327 1032 1 format sequence. DECODE_PROCESS_FORMAT should then be modified to scan
328 1033 1 for the end of the formatting sequence setting any appropriate flags.
329 1034 1 Information to be communicated to the output conversion routine
330 1035 1 (OUTPUT_ARG) is passed via PARAM_BLK. The paramter block could
331 1036 1 be expanded to allow for any new flags.
332 1037 1
333 1038 1
334 1039 1 CLASS_TABLE = UPLIT BYTE
335 1040 1 (
336 1041 1 REP CHAR EXCLAMATION OF (0),
337 1042 1 (MASK_FORMAT_CHARS), ! CHAR EXCLAMATION
338 1043 1 REP CHAR POUND - CHAR_EXCLAMATION - 1 OF (0),
339 1044 1 (MASK_FORMAT_CHARS), ! CHAR POUND
340 1045 1 REP CHAR DOLLAR - CHAR_POUND - 1 OF (0),
341 1046 1 (MASK_FORMAT_CHARS), ! CHAR DOLLAR
342 1047 1 REP CHAR PERCENT - CHAR_DOLLAR - 1 OF (0),
343 1048 1 (MASK_LF_ANGLE), ! CHAR PERCENT
344 1049 1 REP CHAR_SINGLE_QUOTE - CHAR_PERCENT - 1 OF (0),
345 1050 1 (MASK_FORMAT_CHARS), ! CHAR_SINGLE_QUOTE
346 1051 1 REP CHAR_STAR - CHAR_SINGLE_QUOTE - 1 OF (0),
347 1052 1 (MASK_FORMAT_CHARS), ! CHAR_STAR
348 1053 1 REP CHAR_COMMA - CHAR_STAR - 1 OF (0),
349 1054 1 (MASK_SPECIAL), ! CHAR_COMMA

```

```

350 1055 1 REP CHAR MINUS - CHAR_COMMA - 1 OF (0),
351 1056 1 (MASK SPECIAL), ! CHAR_MINUS
352 1057 1 REP CHAR PERIOD - CHAR_MINUS - 1 OF (0),
353 1058 1 (MASK FORMAT CHARS), ! CHAR_PERIOD
354 1059 1 REP CHAR_ZERO - CHAR_PERIOD - 1 OF (0),
355 1060 1 (MASK LF_ANGLE), ! CHAR_ZERO
356 1061 1 REP CHAR_LF_ANGLE - CHAR_ZERO - 1 OF (0),
357 1062 1 (MASK FORMAT CHARS), ! CHAR_LF_ANGLE
358 1063 1 REP CHAR_C - CHAR_LF_ANGLE - 1 OF (0),
359 1064 1 (MASK C), ! CHAR_C
360 1065 1 REP CHAR_E - CHAR_C - 1 OF (0),
361 1066 1 (MASK E), ! CHAR_E
362 1067 1 REP CHAR_L - CHAR_E - 1 OF (0),
363 1068 1 (MASK L), ! CHAR_L
364 1069 1 REP CHAR_R - CHAR_L - 1 OF (0),
365 1070 1 (MASK R), ! CHAR_R
366 1071 1 REP CHAR_BACKSLASH - CHAR_R - 1 OF (0),
367 1072 1 (MASK FORMAT CHARS), ! CHAR_BACKSLASH
368 1073 1 REP CHAR_CARAT - CHAR_BACKSLASH - 1 OF (0),
369 1074 1 (MASK SPECIAL), ! CHAR_CARAT
370 1075 1 REP CHAR_UNDERSCORE - CHAR_CARAT - 1 OF (0),
371 1076 1 (MASK FORMAT CHARS), ! CHAR_UNDERSCORE
372 1077 1 REP CHAR_LOWER_C - CHAR_UNDERSCORE - 1 OF (0),
373 1078 1 (MASK C), ! CHAR_LOWER_C
374 1079 1 REP CHAR_LOWER_E - CHAR_LOWER_C - 1 OF (0),
375 1080 1 (MASK E), ! CHAR_LOWER_E
376 1081 1 REP CHAR_LOWER_L - CHAR_LOWER_E - 1 OF (0),
377 1082 1 (MASK L), ! CHAR_LOWER_L
378 1083 1 REP CHAR_LOWER_R - CHAR_LOWER_L - 1 OF (0),
379 1084 1 (MASK R), ! CHAR_LOWER_R
380 1085 1 REP 255 - CHAR LOWER R OF (0)
381 1086 1 ) : VECTOR [256, BYTE, UNSIGNED];
382 1087 1
383 1088 1
384 1089 1 ! EXTERNAL REFERENCES:
385 1090 1 !
386 1091 1
387 1092 1 EXTERNAL ROUTINE
388 1093 1 LIB$CURRENCY, ! currency symbol
389 1094 1 LIB$RADIX_POINT, ! radix point
390 1095 1 LIB$DIGIT_SEP, ! digit group separator
391 1096 1 BASS$RSET : NOVALUE, ! Basic RSET
392 1097 1 STR$GET1_DX, ! Allocate a string
393 1098 1 STR$FREE1_DX, ! Free a string
394 1099 1 STR$CONCAT, ! Concatenate strings
395 1100 1 STR$RIGHT, ! Extract right part of a string
396 1101 1 STR$LEN_EXTR, ! Extract a substring
397 1102 1 LIB$STOP, ! Signal internal RTL errors
398 1103 1 STR$COPY_DX R8 : STR$JSB_COPY_DX, ! Copy to a string
399 1104 1 STR$COPY_R R8 : STR$JSB_COPY_R, ! Copy to a dynamic string
400 1105 1 BASS$STOP : NOVALUE, ! Signal errors and stop
401 1106 1 BASS$CVT_OUT_F_E, ! convert float to E format
402 1107 1 BASS$CVT_OUT_D_E, ! convert double to E format
403 1108 1 BASS$CVT_OUT_F_F, ! convert float to F format
404 1109 1 BASS$CVT_OUT_D_F, ! convert double to F format
405 1110 1 BASS$CVT_OUT_D_G, ! PRINT format conversion routine
406 1111 1 BASS$CVT_OUT_G_G, ! PRINT format conversion routine

```

```

: 407      1112 1      BASS$CVT_OUT_G_E,      : convert gfloat to E format
: 408      1113 1      BASS$CVT_OUT_G_F,      : convert gfloat to F format
: 409      1114 1      BASS$CVT_OUT_H_E,      : convert hfloat to E format
: 410      1115 1      BASS$CVT_OUT_H_G,      : PRINT format conversion routine
: 411      1116 1      BASS$CVT_OUT_H_F,      : convert hfloat to F format
: 412      1117 1      BASS$CVT_OUT_P_F,      : convert packed to F format
: 413      1118 1      BASS$CVT_OUT_P_E,      : convert packed to E format
: 414      1119 1      BASS$CVT_OUT_P_G;      : convert packed to general format
: 415      1120 1
: 416      1121 1      EXTERNAL LITERAL
: 417      1122 1      BASS$K_PRIUSIFOR,      ! Print Using format error
: 418      1123 1      OTSS_$ATINTERR;      ! Fatal internal RTL error
: 419      1124 1
: 420      1125 1      BUILTIN
: 421      1126 1
: 422      1127 1      !+
: 423      1128 1      ! In version X3-669 of VAX-11 Bliss-32, output registers will be allowed
: 424      1129 1      ! for the SCANC and SPANC instructions (there is a bug in V2.1-667 that
: 425      1130 1      ! prevents the use of output registers with SCANC and SPANC or
: 426      1131 1      ! any BUILTIN for that matter). These output
: 427      1132 1      ! registers could be used to return information from the instructions
: 428      1133 1      ! that currently requires a computation after the BUILTIN has been executed.
: 429      1134 1
: 430      1135 1      ! In version V3 of VAX-11 Bliss-32, LOCC and SKPC will be allowed as BUILTINS
: 431      1136 1      ! so that the use of CH$FIND_NOT_CH could be replaced by the BUILTIN SKPC
: 432      1137 1      ! with output registers.
: 433      1138 1      !-
: 434      1139 1
: 435      1140 1      SCANC,
: 436      1141 1      SPANC,
: 437      1142 1      CVTLD;
: 438      1143 1

```

```

: 440      1144 1 %SBTTL 'BASS$FORMAT_INT - external entry point'
: 441      1145 1 GLOBAL ROUTINE BASS$FORMAT_INT (           ! Basic format interpreter
: 442      1146 1
: 443      1147 1     ELEMENT,           ! element to format
: 444      1148 1     FORMAT_STR,       ! string containing the formatting information
: 445      1149 1     ELEMENT_TYPE,    ! data type of the element
: 446      1150 1     RET_STRING,      ! where to return the result
: 447      1151 1     RET_FORMAT_ADR,   ! address of the next char in format string
: 448      1152 1     BAS_SCALE_FAC    ! Scale factor (-6 <= x <= 0)
: 449      1153 1     ) : NOVALUE =
: 450      1154 1
: 451      1155 2 BEGIN
: 452      1156 2
: 453      1157 2 !++
: 454      1158 2 ! FUNCTIONAL DESCRIPTION:
: 455      1159 2
: 456      1160 2     This is the format interpreter for Basic Print Using and FORMATS.
: 457      1161 2     It will interpret the format string and call the appropriate output
: 458      1162 2     conversion routine.
: 459      1163 2
: 460      1164 2     Note that this routine will be called for each item in the format
: 461      1165 2     list. Thus if there are 10 items in the format list and 1 formatting
: 462      1166 2     sequence, this routine will be called once for each item (10 times).
: 463      1167 2     If the PRINT USING is in a loop of 1,000 iterations, this routine
: 464      1168 2     will be called once for each item-iteration or 10,000 times
: 465      1169 2     in the above example. Thus the format sequence will be decoded
: 466      1170 2     10,000 times. There is obvious room for improvement here by having
: 467      1171 2     a separate routine to set up the PRINT USING parameter block with
: 468      1172 2     the formatting information. Note that the parameter block
: 469      1173 2     currently used only handles one format sequence at a time.
: 470      1174 2     Some higher level caller would have to save however many decoded
: 471      1175 2     PRINT USING parameter blocks might be needed.
: 472      1176 2
: 473      1177 2     The routines in this module would benefit from RTL routines that
: 474      1178 2     provide STR$APPEND with multiple source string descriptors and
: 475      1179 2     STR$APPEND_R capabilities to append the source string by reference.
: 476      1180 2
: 477      1181 2     The PRINT USING support implemented here provides for upper
: 478      1182 2     and lower case text formatting characters (L,C,R,E and l,c,r,e).
: 479      1183 2     PDP-11 BASIC-PLUS-2 only allows upper case - lower case letters
: 480      1184 2     (l,c,r,e) are treated as constants. Thus the capabilities provided
: 481      1185 2     here are not strictly a superset of the PDP-11 capabilities.
: 482      1186 2
: 483      1187 2 FORMAL PARAMETERS:
: 484      1188 2
: 485      1189 2     ELEMENT.rx.r      element to format
: 486      1190 2     FORMAT_STR.rt.dx    string containing the formatting information
: 487      1191 2     Note that throughout this module, '.dx' is used
: 488      1192 2     for FORMAT_STR where in fact it only supports
: 489      1193 2     '.dd' and '.ds'. We need a notation to specify
: 490      1194 2     '.dd' and '.ds'.
: 491      1195 2     ELEMENT_TYPE.rl.v    data type of the element
: 492      1196 2     RET_STRING.wt.dd     where to return the result
: 493      1197 2     RET_FORMAT_ADR.wl.r  address of next char in format string
: 494      1198 2     BAS_SCALE_FAC.rb.v   Scale factor (-6 <= x <= 0)
: 495      1199 2
: 496      1200 2 ! IMPLICIT INPUTS:

```



```

554 1258 3 !-
555 1259 3 DIGIT_SEP_STR [DSC$W_LENGTH] = 0;
556 1260 3 DIGIT_SEP_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
557 1261 3 DIGIT_SEP_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
558 1262 3 DIGIT_SEP_STR [DSC$A_POINTER] = 0;
559 1263 3 RADIX_PT_STR [DSC$W_LENGTH] = 0;
560 1264 3 RADIX_PT_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
561 1265 3 RADIX_PT_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
562 1266 3 RADIX_PT_STR [DSC$A_POINTER] = 0;
563 1267 3 CURRENCY_STR [DSC$W_LENGTH] = 0;
564 1268 3 CURRENCY_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
565 1269 3 CURRENCY_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
566 1270 3 CURRENCY_STR [DSC$A_POINTER] = 0;
567 1271 3
568 1272 4 BEGIN
569 1273 4 MAP
570 1274 4 RADIX_PT_STR : VECTOR [2, LONG],
571 1275 4 DIGIT_SEP_STR : VECTOR [2, LONG],
572 1276 4 CURRENCY_STR : VECTOR [2, LONG];
573 1277 4 LIB$CURRENCY(CURRENCY_STR);
574 1278 4
575 1279 4 !+
576 1280 4 Disable ASTs to guard against an AST right here which does a PRINT USING or
577 1281 4 a FORMATS which allocates another string resulting in its being lost forever.
578 1282 4
579 1283 4 The disabling of ASTs will be controlled by K_AST_FLAG.
580 1284 4 !-
581 1285 4
582 1286 4 IF K_AST_FLAG
583 1287 4 THEN
584 1288 5 BEGIN
585 1289 5 AST_STATUS = $SETAST(ENBFLG = 0);
586 1290 5 CURRENCY_DESC = .CURRENCY_STR [0];
587 1291 5 CU_D = .CURRENCY_STR [1];
588 1292 5
589 1293 5 !+
590 1294 5 Re-enable ASTs if they were previously.
591 1295 5 !-
592 1296 5
593 1297 5 IF (.AST_STATUS EQL S$$_WASSET) THEN $SETAST(ENBFLG = 1);
594 1298 5 LIB$RADIX_POINT(RADIX_PT_STR);
595 1299 5 $SETAST(ENBFLG = 0);
596 1300 5 RADIX_PT_DESC = .RADIX_PT_STR [0];
597 1301 5 RA_D = .RADIX_PT_STR [1];
598 1302 5 IF (.AST_STATUS EQL S$$_WASSET) THEN $SETAST(ENBFLG = 1);
599 1303 5 LIB$DIGIT_SEP(DIGIT_SEP_STR);
600 1304 5 $SETAST(ENBFLG = 0);
601 1305 5 DIGIT_SEP_DESC = .DIGIT_SEP_STR [0];
602 1306 5 DI_D = .DIGIT_SEP_STR [1];
603 1307 5 IF (.AST_STATUS EQL S$$_WASSET) THEN $SETAST(ENBFLG = 1);
604 1308 5 END
605 1309 5
606 1310 4 ELSE
607 1311 5 BEGIN
608 1312 5 CURRENCY_DESC = .CURRENCY_STR [0];
609 1313 5 CU_D = .CURRENCY_STR [1];
610 1314 5 LIB$RADIX_POINT(RADIX_PT_STR);

```

```

: 611      1315  S      RADIX_PT_DESC = .RADIX_PT_STR [0];
: 612      1316  S      RA_D = .RADIX_PT_STR [1];
: 613      1317  S      LIB$DIGIT_SEPT(DIGIT_SEP_STR);
: 614      1318  S      DIGIT_SEP_DESC = .DIGIT_SEP_STR [0];
: 615      1319  S      DI_D = .DIGIT_SEP_STR [1]
: 616      1320  S      END;
: 617      1321  S
: 618      1322  S      END;
: 619      1323  S      END;
: 620      1324  S
: 621      1325  S
: 622      1326  S      + Initialize the parameter block and the data base that Print Using
: 623      1327  S      will use.
: 624      1328  S
: 625      1329  S
: 626      1330  S      PARAM_BLK [INTEGER_DIGITS] = PARAM_BLK [CHARACTER] = PARAM_BLK [FRACTION_DIGIT] =
: 627      1331  S      PARAM_BLK [PU_MASK] = 0;
: 628      1332  S      PARAM_BLK [RET_STR] = .RET_STRING;
: 629      1333  S      PARAM_BLK [ELEM] = .ELEMENT;
: 630      1334  S      PARAM_BLK [ELEM_TYPE] = .ELEMENT_TYPE;
: 631      1335  S      PARAM_BLK [SCALE_FACTOR] = .BAS_SCALE_FAC [0];
: 632      1336  S
: 633      1337  S
: 634      1338  S      + The format decoding routine takes over here and calls the output
: 635      1339  S      conversion routine after setting up PARAM_BLK.
: 636      1340  S
: 637      1341  S
: 638      1342  S      IF NOT (DECODE_PROCESS_FORMAT (.FORMAT_STR, PARAM_BLK))
: 639      1343  S      THEN
: 640      1344  S
: 641      1345  S
: 642      1346  S      + Signal a fatal internal error in the RTL. All errors should be caught
: 643      1347  S      in the routines called by DECODE_PROCESS_FORMAT.
: 644      1348  S
: 645      1349  S
: 646      1350  S      LIB$STOP (OTSS_FATINTERR);
: 647      1351  S
: 648      1352  S      RET_FORMAT_ADR [0] = .PARAM_BLK [NEXT_CHAR];
: 649      1353  S      RETURN;
: 650      1354  S      END;

```

! End of routine BASS\$FORMAT_INT

```

.TITLE BASS$FOR_INT BASS$FORMAT_INT - Basic format int
       erpreter
.IDENT \2-013\
.PSECT _BAS$DATA,NOEXE, PIC,2

```

```

00000 CURRENCY_DESC:
      .BLKB 4
00004 CU_D: .BLKB 4
00008 DIGIT_SEP_DESC:
      .BLKB 4
0000C DI_D: .BLKB 4
00010 RADIX_PT_DESC:
      .BLKB 4
00014 RA_D: .BLKB 4

```

```

.PSECT _BAS$CODE,NOWRT, SHR, PIC,2
SE# 00000 P.AAA: .BYTE 94[4]
25 00004 P.AAB: .ASCII \X\
00# 00005 P.AAC: .BYTE 0[33]
01 00026 .BYTE 1
00 00027 .BYTE 0
01 00028 .BYTE 1
01 00029 .BYTE 1
40 0002A .BYTE 64
00 0002B .BYTE 0
01 0002C .BYTE 1
00# 0002D .BYTE 0[2]
01 0002F .BYTE 1
00 00030 .BYTE 0
20 00031 .BYTE 32
20 00032 .BYTE 32
01 00033 .BYTE 1
00 00034 .BYTE 0
40 00035 .BYTE 64
00# 00036 .BYTE 0[11]
01 00041 .BYTE 1
00# 00042 .BYTE 0[6]
04 00048 .BYTE 4
00 00049 .BYTE 0
10 0004A .BYTE 16
00# 0004B .BYTE 0[6]
02 00051 .BYTE 2
00# 00052 .BYTE 0[5]
08 00057 .BYTE 8
00# 00058 .BYTE 0[9]
01 00061 .BYTE 1
00 00062 .BYTE 0
20 00063 .BYTE 32
01 00064 .BYTE 1
00# 00065 .BYTE 0[3]
04 00068 .BYTE 4
00 00069 .BYTE 0
10 0006A .BYTE 16
00# 0006B .BYTE 0[6]
02 00071 .BYTE 2
00# 00072 .BYTE 0[5]
08 00077 .BYTE 8
00# 00078 .BYTE 0[141]

```

```

E_FORMAT= P.AAA
PERCENT= P.AAB
CLASS_TABLE= P.AAC
.EXTRN LIB$CURRENCY, LIB$RADIX_POINT
.EXTRN LIB$DIGIT_SEP, BAS$RSET
.EXTRN STR$GET1_DX, STR$FREE1_DX
.EXTRN STR$CONCAT, STR$RIGHT
.EXTRN STR$LEN_EXTR, LIB$STOP
.EXTRN STR$COPY_DX_R8, STR$COPY_R_R8
.EXTRN BAS$$STOP, BAS$CVT_OUT_F_E
.EXTRN BAS$CVT_OUT_D_E

```

| | | | | | | | | |
|--|--|--|------------|--|--------|---|------|------|
| | | | | | .EXTRN | BAS\$CVT_OUT_F_F | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_D_F | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_D_G | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_G_G | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_G_E | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_G_F | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_H_E | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_H_G | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_H_F | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_P_F | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_P_E | | |
| | | | | | .EXTRN | BAS\$CVT_OUT_P_G | | |
| | | | | | .EXTRN | BAS\$K_PRIUSTFOR | | |
| | | | | | .EXTRN | OT\$\$_FATINTERR, SYS\$SETAST | | |
| | | | 07FC 00000 | | .ENTRY | BAS\$\$FORMAT_INT, Save R2,R3,R4,R5,R6,R7,R8,-; | 1145 | |
| | | | | | | R9,R10 | | |
| | | | | | MOVAB | RA_D, R10 | | |
| | | | | | MOVAB | SYS\$SETAST, R9 | | |
| | | | | | SUBL2 | #60, SP | | |
| | | | | | CLRL | -(SP) | | 1241 |
| | | | | | MOVAB | (SP), R2 | | |
| | | | | | CLRL | R1 | | |
| | | | | | MOVL | RET_STRING, R0 | | |
| | | | | | JSB | STR\$COPY_R_R8 | | |
| | | | | | TSTL | RA_D | | 1248 |
| | | | | | BNEQ | 3\$ | | |
| | | | | | MOVL | #34471936, DIGIT_SEP_STR | | 1259 |
| | | | | | CLRL | DIGIT_SEP_STR+4 | | 1262 |
| | | | | | MOVL | #34471936, RADIX_PT_STR | | 1263 |
| | | | | | CLRL | RADIX_PT_STR+4 | | 1266 |
| | | | | | MOVL | #34471936, CURRENCY_STR | | 1267 |
| | | | | | CLRL | CURRENCY_STR+4 | | 1270 |
| | | | | | PUSHAB | CURRENCY_STR | | 1277 |
| | | | | | CALLS | #1, LIB\$CURRENCY | | |
| | | | | | CLRL | -(SP) | | 1289 |
| | | | | | CALLS | #1, SYS\$SETAST | | |
| | | | | | MOVQ | CURRENCY_STR, CURRENCY_DESC | | 1290 |
| | | | | | CLRL | R2 | | 1297 |
| | | | | | CMP | AST_STATUS, #9 | | |
| | | | | | BNEQ | 1\$ | | |
| | | | | | INCL | R2 | | |
| | | | | | PUSHL | #1 | | |
| | | | | | CALLS | #1, SYS\$SETAST | | |
| | | | | | PUSHAB | RADIX_PT_STR | | 1298 |
| | | | | | CALLS | #1, LIB\$RADIX_POINT | | |
| | | | | | CLRL | -(SP) | | 1299 |
| | | | | | CALLS | #1, SYS\$SETAST | | |
| | | | | | MOVQ | RADIX_PT_STR, RADIX_PT_DESC | | 1300 |
| | | | | | BLBC | R2, 2\$ | | 1302 |
| | | | | | PUSHL | #1 | | |
| | | | | | CALLS | #1, SYS\$SETAST | | |
| | | | | | PUSHAB | DIGIT_SEP_STR | | 1303 |
| | | | | | CALLS | #1, LIB\$DIGIT_SEP | | |
| | | | | | CLRL | -(SP) | | 1304 |
| | | | | | CALLS | #1, SYS\$SETAST | | |
| | | | | | MOVQ | DIGIT_SEP_STR, DIGIT_SEP_DESC | | 1305 |

BAS\$\$FOR_INT
2-013

BAS\$\$FORMAT_INT - Basic format interpreter
BAS\$\$FORMAT_INT - external entry point

F 6
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 16
(3)

| | | | | | | | | | |
|-----------|----|-----------|----|-------|-------|--------|-------------------------------|---|------|
| | 05 | | 52 | E9 | 0009B | BLBC | R2, 3\$ | : | 1307 |
| | | | 01 | DD | 0009E | PUSHL | #1 | : | |
| | 69 | | 01 | FB | 000A0 | CALLS | #1, SY\$\$SETAST | : | |
| | | 20 | AE | D4 | 000A3 | CLRL | PARAM_BLK+4 | : | 1331 |
| | | 34 | AE | D4 | 000A6 | CLRL | PARAM_BLK+24 | : | 1330 |
| | | 2C | AE | 7C | 000A9 | CLRQ | PARAM_BLK+16 | : | 1331 |
| 1C | AE | 10 | AC | D0 | 000AC | MOVL | RET STRING, PARAM_BLK | : | 1332 |
| 28 | AE | 04 | AC | D0 | 000B1 | MOVL | ELEMENT, PARAM_BLK+12 | : | 1333 |
| 24 | AE | 0C | AC | D0 | 000B6 | MOVL | ELEMENT TYPE, PARAM_BLK+8 | : | 1334 |
| 38 | AE | 18 | AC | 98 | 000BB | CVTBL | BAS SCALE_FAC, PARAM_BLK+28 | : | 1335 |
| | | 1C | AE | 9F | 000C0 | PUSHAB | PARAM_BLK | : | 1342 |
| | | 08 | AC | DD | 000C3 | PUSHL | FORMAT_STR | : | |
| 0000V | CF | | 02 | FB | 000C6 | CALLS | #2, DECODE_PROCESS_FORMAT | : | |
| | 0D | | 50 | E8 | 000CB | BLBS | R0, 4\$ | : | |
| | | 00000000G | 8F | DD | 000CE | PUSHL | #0T\$\$ FATINTERR | : | 1350 |
| 00000000G | 00 | | 01 | FB | 000D4 | CALLS | #1, LIB\$\$STOP | : | |
| 14 | BC | 3C | AE | D0 | 000DB | MOVL | PARAM_BLK+32, @RET_FORMAT_ADR | : | 1352 |
| | | | 04 | 000E0 | RET | | | : | 1354 |

; Routine Size: 225 bytes, Routine Base: _BAS\$CODE + 0105

; 651 1355 1

```

: 653 1356 1 XSBTTL 'DECODE_PROCESS_FORMAT - main driving routine'
: 654 1357 1 ROUTINE DECODE_PROCESS_FORMAT (
: 655 1358 1
: 656 1359 1     P_FORMAT_STR,  ! format string
: 657 1360 1     P_PARAM_BLK) = ! print using parameter block
: 658 1361 1
: 659 1362 2 BEGIN
: 660 1363 2
: 661 1364 2 ++
: 662 1365 2
: 663 1366 2 FUNCTIONAL DESCRIPTION:
: 664 1367 2
: 665 1368 2     The plan of attack is to locate the start of the formatting sequence.
: 666 1369 2     All characters up to the formatting sequence are concatenated onto
: 667 1370 2     the return string. A parameter block is then set up based on the
: 668 1371 2     formatting information and OUTPUT_ARG is called to do the
: 669 1372 2     actual formatting concatenating the result to the return string.
: 670 1373 2     The beginning of the next formatting sequence is then searched for
: 671 1374 2     and all characters up to this next formatting sequence are concatenated
: 672 1375 2     onto the return string.
: 673 1376 2
: 674 1377 2     The SPAN_CONSTANT routine will position CHAR_CNT at the last character
: 675 1378 2     before the formatting sequence. CHAR_CNT = 0 implies that no leading
: 676 1379 2     constant was found. The constant string up to the formatting sequence
: 677 1380 2     is copied onto the return string. If no formatting sequence
: 678 1381 2     is found, 0 is returned by SPAN_CONSTANT and a print using format error
: 679 1382 2     is signalled; otherwise 1 is returned by SPAN_CONSTANT.
: 680 1383 2
: 681 1384 2     After the leading constant is located, the proper formatting sequence
: 682 1385 2     is located and process. Then the trailing constant is processed.
: 683 1386 2
: 684 1387 2     The calling structure of the routines is as follows:
: 685 1388 2
: 686 1389 2     BAS$$FORMAT_INT
: 687 1390 2     DECODE_PROCESS_FORMAT
: 688 1391 2     SPAN_CONSTANT
: 689 1392 2     SPAN_SINGLE_QUOTE
: 690 1393 2     SPAN_TEXT
: 691 1394 2     SPAN_BACKSLASH
: 692 1395 2     SPAN_NUMERIC
: 693 1396 2     OUTPUT_ARG
: 694 1397 2     SPAN_CONSTANT
: 695 1398 2
: 696 1399 2 CALLING SEQUENCE:
: 697 1400 2
: 698 1401 2     STATUS.wlc.v = DECODE_PROCESS_FORMAT (P_FORMAT_STR.rt.dx,
: 699 1402 2     P_PARAM_BLK.mr.r)
: 700 1403 2
: 701 1404 2 FORMAL PARAMETERS:
: 702 1405 2
: 703 1406 2     P_FORMAT_STR - address of string descriptor for format string
: 704 1407 2     P_PARAM_BLK - address of print using parameter block
: 705 1408 2
: 706 1409 2 IMPLICIT INPUTS:
: 707 1410 2
: 708 1411 2     CURRENCY_DESC - descriptor for currency symbol
: 709 1412 2

```

```

: 710      1413 2 | IMPLICIT OUTPUTS:
: 711      1414 2 |
: 712      1415 2 |     NONE
: 713      1416 2 |
: 714      1417 2 | COMPLETION CODES:
: 715      1418 2 |
: 716      1419 2 |     1           - routine completed successfully
: 717      1420 2 |     0           - error
: 718      1421 2 |
: 719      1422 2 | SIDE EFFECTS:
: 720      1423 2 |
: 721      1424 2 |     NONE
: 722      1425 2 |
: 723      1426 2 | --
: 724      1427 2 |
: 725      1428 2 | BIND
: 726      1429 2 |     PARAM_BLK = (.P_PARAM_BLK) : BLOCK [, BYTE],
: 727      1430 2 |     FORMAT_STR = (.P_FORMAT_STR) : BLOCK [, BYTE];
: 728      1431 2 |
: 729      1432 2 | MAP
: 730      1433 2 |     CURRENCY_DESC : BLOCK [, BYTE];
: 731      1434 2 |
: 732      1435 2 | LOCAL
: 733      1436 2 |     CHAR_CNT;      ! keeps track of offset from beginning
: 734      1437 2 |                    ! of format string
: 735      1438 2 |
: 736      1439 2 | CHAR_CNT = .FORMAT_STR [DSC$W_LENGTH];
: 737      1440 2 | CHAR_CNT = 0;
: 738      1441 2 |
: 739      1442 2 | !+
: 740      1443 2 | ! SPAN_CONSTANT returns 0 if there is no format sequence or a null format
: 741      1444 2 | ! string.
: 742      1445 2 | !-
: 743      1446 2 |
: 744      1447 2 | IF NOT SPAN_CONSTANT (FORMAT_STR, .PARAM_BLK [RET_STR], CHAR_CNT)
: 745      1448 2 | THEN
: 746      1449 2 |     BEGIN
: 747      1450 2 |     BAS$$STOP (BAS$K_PRIUSIFOR);
: 748      1451 2 |     RETURN 1           ! return if error
: 749      1452 2 |     END
: 750      1453 2 |
: 751      1454 2 | ELSE
: 752      1455 2 |     BEGIN
: 753      1456 2 |
: 754      1457 2 | !+
: 755      1458 2 | ! Decode and process formatting sequence.
: 756      1459 2 | !-
: 757      1460 2 |
: 758      1461 2 |     BIND CHAR = (.FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT) : BYTE;
: 759      1462 2 |
: 760      1463 2 |     SELECTONEU .CHAR OF
: 761      1464 2 |     SET
: 762      1465 2 |
: 763      1466 2 |     [CHAR_SINGLE_QUOTE] :
: 764      1467 2 |     BEGIN
: 765      1468 2 |     CHAR_CNT = .CHAR_CNT + 1;
: 766      1469 2 |     SPAN_SINGLE_QUOTE (FORMAT_STR, PARAM_BLK, CHAR_CNT);

```

```

: 767      1470  4      OUTPUT_ARG (PARAM_BLK);
: 768      1471  4      !+
: 769      1472  4      ! If there is an underscore in the text format string, the next character
: 770      1473  4      ! should be appended into a trailing constant. Any other characters which
: 771      1474  4      ! should be part of the trailing constant will be handled in the next call
: 772      1475  4      ! to SPAN_CONSTANT.
: 773      1476  4      !-
: 774      1477  4      SPAN_CONSTANT (FORMAT_STR, .PARAM_BLK [RET_STR], CHAR_CNT)
: 775      1478  4      END;
: 776      1479  4
: 777      1480  3      [CHAR_POUND] :
: 778      1481  4      BEGIN
: 779      1482  4      CHAR_CNT = .CHAR_CNT + 1;
: 780      1483  4      PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR NUMERIC_MASK;
: 781      1484  4      PARAM_BLK [INTEGER_DIGITS] = 1;
: 782      1485  4      SPAN_NUMERIC (K SPAN_INTEGER, FORMAT_STR,
: 783      1486  4      CHAR_CNT, PARAM_BLK);
: 784      1487  4      OUTPUT_ARG (PARAM_BLK)
: 785      1488  4      END;
: 786      1489  4
: 787      1490  4      [CHAR_DOLLAR] :          ! floating currency
: 788      1491  4
: 789      1492  4      !+
: 790      1493  4      ! The SPAN_CONSTANT routine will have insured that the intial part of the
: 791      1494  4      ! formatting sequence is '$$'.
: 792      1495  4      !-
: 793      1496  4
: 794      1497  4      BEGIN
: 795      1498  4      CHAR_CNT = .CHAR_CNT + 2;
: 796      1499  4      PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR DOLLAR_MASK
: 797      1500  4      OR NUMERIC_MASK;
: 798      1501  4      PARAM_BLK [INTEGER_DIGITS] = 1 + .CURRENCY_DESC [DSCSW_LENGTH];
: 799      1502  4      !+
: 800      1503  4      ! A '<%' or '<0>' could follow '$$'. Check for this before scanning the rest
: 801      1504  4      ! of the format string. Don't check for <0> anymore as of edit 2-012 to be
: 802      1505  4      ! compatible with BP2.
: 803      1506  4      !-
: 804      1507  4      SPAN_BRACKET (CHAR_PERCENT, FORMAT_STR, PARAM_BLK, CHAR_CNT);
: 805      1508  4      SPAN_BRACKET (CHAR_ZERO, FORMAT_STR, PARAM_BLK, CHAR_CNT);
: 806      1509  4      SPAN_NUMERIC (K SPAN_INTEGER_NO_E, FORMAT_STR,
: 807      1510  4      CHAR_CNT, PARAM_BLK);
: 808      1511  4      OUTPUT_ARG (PARAM_BLK)
: 809      1512  4      END;
: 810      1513  4
: 811      1514  4      [CHAR_STAR] :          ! asterisk fill
: 812      1515  4
: 813      1516  4      !+
: 814      1517  4      ! The SPAN_CONSTANT routine will have insured that the intial part of the
: 815      1518  4      ! formatting sequence is '**'.
: 816      1519  4      !-
: 817      1520  4
: 818      1521  4      BEGIN
: 819      1522  4      CHAR_CNT = .CHAR_CNT + 2;
: 820      1523  4      PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR STAR_MASK
: 821      1524  4      OR NUMERIC_MASK;
: 822      1525  4      PARAM_BLK [INTEGER_DIGITS] = 2;
: 823      1526  4      !+

```

```

: 824 1527 4 | A '<%>' could follow '**'. Check for this before scanning the rest of the
: 825 1528 4 | format string.
: 826 1529 4 | -
: 827 1530 4 | SPAN_BRACKET (CHAR_PERCENT, FORMAT_STR, PARAM_BLK, CHAR_CNT);
: 828 1531 4 | SPAN_NUMERIC (K_SPAN_INTEGER_NO_E, FORMAT_STR,
: 829 1532 4 | CHAR_CNT, PARAM_BLK);
: 830 1533 4 | OUTPUT_ARG (PARAM_BLK)
: 831 1534 3 | END;
: 832 1535 3 |
: 833 1536 3 | [CHAR_PERIOD] :
: 834 1537 4 | BEGIN
: 835 1538 4 |
: 836 1539 4 | !+ The SPAN_CONSTANT routine will have insured that the intial part of the
: 837 1540 4 | formatting sequence is '.#'.
: 838 1541 4 | -
: 839 1542 4 |
: 840 1543 4 |
: 841 1544 4 | CHAR_CNT = .CHAR_CNT + 2;
: 842 1545 4 | PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR PERIOD_MASK
: 843 1546 4 | OR NUMERIC_MASK;
: 844 1547 4 | PARAM_BLK [FRACTION_DIGIT] = 1;
: 845 1548 4 | SPAN_NUMERIC (K_SPAN_FRACTION, FORMAT_STR,
: 846 1549 4 | CHAR_CNT, PARAM_BLK);
: 847 1550 4 | OUTPUT_ARG (PARAM_BLK);
: 848 1551 3 | END;
: 849 1552 3 |
: 850 1553 3 | [CHAR_BACKSLASH] :
: 851 1554 4 | BEGIN
: 852 1555 4 | CHAR_CNT = .CHAR_CNT + 1;
: 853 1556 4 | IF NOT SPAN_BACKSLASH (FORMAT_STR, PARAM_BLK, CHAR_CNT)
: 854 1557 4 | THEN
: 855 1558 5 | BEGIN
: 856 1559 5 | BASS$STOP (BASS$PRIUSIFOR);
: 857 1560 5 | RETURN 1; ! return if error
: 858 1561 5 | END
: 859 1562 5 |
: 860 1563 4 | ELSE
: 861 1564 4 | OUTPUT_ARG (PARAM_BLK)
: 862 1565 4 |
: 863 1566 3 | END;
: 864 1567 3 |
: 865 1568 3 | [CHAR_EXCLAMATION] :
: 866 1569 4 | BEGIN
: 867 1570 4 | CHAR_CNT = .CHAR_CNT + 1;
: 868 1571 4 | PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK;
: 869 1572 4 | PARAM_BLK [CHARACTER] = 1;
: 870 1573 4 | OUTPUT_ARG (PARAM_BLK)
: 871 1574 3 | END;
: 872 1575 3 |
: 873 1576 3 | !+ The SPAN_CONSTANT routine has insured that a valid '< >' sequence
: 874 1577 3 | is present.
: 875 1578 3 | -
: 876 1579 3 |
: 877 1580 3 |
: 878 1581 3 | [CHAR_LF_ANGLE] :
: 879 1582 4 | BEGIN
: 880 1583 4 | BIND CHAR2 = (.FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT + 1) : BYTE;

```

```

881      1584      4
882      1585      4      SELECTONEU .CHAR2 OF
883      1586      4      SET
884      1587      4      [CHAR PERCENT]:
885      1588      4      BEGIN
886      1589      4      CHAR CNT = .CHAR CNT + 3;
887      1590      4      PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR PERCENT_MASK
888      1591      4      OR NUMERIC_MASK;
889      1592      4      PARAM_BLK [INTEGER DIGITS] = .PARAM_BLK [INTEGER DIGITS] + 1;
890      1593      4      SPAN_NUMERIC (K_SPAN_INTEGER, FORMAT_STR, CHAR_CNT,
891      1594      4      PARAM_BLK);
892      1595      4      OUTPUT_ARG (PARAM_BLK)
893      1596      4      END;
894      1597      4
895      1598      4      [CHAR ZERO]:
896      1599      4      BEGIN
897      1600      4      CHAR CNT = .CHAR CNT + 3;
898      1601      4      PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR ZERO_MASK
899      1602      4      OR NUMERIC_MASK;
900      1603      4      PARAM_BLK [INTEGER DIGITS] = .PARAM_BLK [INTEGER DIGITS] + 1;
901      1604      4      SPAN_NUMERIC (K_SPAN_INTEGER, FORMAT_STR, CHAR_CNT,
902      1605      4      PARAM_BLK);
903      1606      4      OUTPUT_ARG (PARAM_BLK)
904      1607      4      END;
905      1608      4
906      1609      4      TES;
907      1610      4      END;
908      1611      4
909      1612      4      [OTHERWISE] :
910      1613      4
911      1614      4      !+
912      1615      4      ! This should never happen.
913      1616      4      !-
914      1617      4
915      1618      4      RETURN 0;          ! return if error (this is a fatal
916      1619      4      ! internal error)
917      1620      4
918      1621      4      TES
919      1622      4      END;
920      1623      4
921      1624      4      !+
922      1625      4      ! At this point, any trailing constant in the format string should be
923      1626      4      ! concatenated onto the return string.
924      1627      4
925      1628      4      ! The return from SPAN_CONSTANT is ignored since we do not care at this
926      1629      4      ! point whether or not the constant string runs to the end of the format
927      1630      4      ! string. In the initial call to SPAN_CONSTANT, we did care because
928      1631      4      ! a null formatting sequence must be signalled as an error. If this
929      1632      4      ! code is executed, a legal formatting sequence has been found and processed.
930      1633      4      ! SPAN_CONSTANT is taking care to get CHAR_CNT to point just before the
931      1634      4      ! next formatting sequence (if one exists).
932      1635      4      !-
933      1636      4
934      1637      4      SPAN_CONSTANT (FORMAT_STR, .PARAM_BLK [RET_STR], CHAR_CNT);
935      1638      4
936      1639      4      !+
937      1640      4      ! Now that the trailing constant has been handled; this routine

```

```

: 938      1641 2 ! has completed its work.
: 939      1642 2 !-
: 940      1643 2
: 941      1644 2 PARAM_BLK [NEXT_CHAR] = .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT;
: 942      1645 2
: 943      1646 2 RETURN 1
: 944      1647 2
: 945      1648 2 END;

```

! End of routine DECODE_PROCESS_FORMAT

| | | 003C 0000 DECODE_PROCESS_FORMAT: | | | | |
|----|-------|----------------------------------|-----------------------|------------|-----------------------|--------|
| | 55 | 0000V | CF 9E 00002 | .WORD | Save R2,R3,R4,R5 | : 1357 |
| | 52 | 08 | AC D0 00007 | MOVAB | SPAN_CONSTANT, R5 | : 1429 |
| | 54 | 04 | AC D0 0000B | MOVL | P_PARAM_BLK, R2 | : 1430 |
| | 7E | | 64 3C 0000F | MOVZWL | (R4), CHAR_CNT | : 1439 |
| | | | 6E D4 00012 | CLRL | CHAR_CNT | : 1440 |
| | | | 5E DD 00014 | PUSHL | SP | : 1447 |
| | | | 62 DD 00016 | PUSHL | (R2) | |
| | | | 54 DD 00018 | PUSHL | R4 | |
| | 65 | | 03 FB 0001A | CALLS | #3, SPAN_CONSTANT | |
| | 03 | | 50 E8 0001D | BLBS | R0, 1\$ | |
| | | | 00AF 31 00020 | BRW | 9\$ | |
| 53 | 04 | A4 | 6E C1 00023 | 1\$: ADDL3 | CHAR_CNT, 4(R4), R3 | : 1461 |
| | | 27 | 63 91 00028 | CMPB | (R3), #39 | : 1466 |
| | | | 20 12 0002B | BNEQ | 2\$ | |
| | | | 6E D6 0002D | INCL | CHAR_CNT | : 1468 |
| | | 4004 | 8F BB 0002F | PUSHR | #*M<R2,SP> | : 1469 |
| | | | 54 DD 00033 | PUSHL | R4 | |
| | 0000V | CF | 03 FB 00035 | CALLS | #3, SPAN_SINGLE_QUOTE | |
| | | | 52 DD 0003A | PUSHL | R2 | : 1470 |
| | 0000V | CF | 01 FB 0003C | CALLS | #1, OUTPUT_ARG | |
| | | | 5E DD 00041 | PUSHL | SP | : 1477 |
| | | | 62 DD 00043 | PUSHL | (R2) | |
| | | | 54 DD 00045 | PUSHL | R4 | |
| | 65 | | 03 FB 00047 | CALLS | #3, SPAN_CONSTANT | |
| | | | 00E4 31 0004A | BRW | 17\$ | |
| | 23 | | 63 91 0004D | 2\$: CMPB | (R3), #35 | : 1480 |
| | | | 0D 12 00050 | BNEQ | 3\$ | |
| | | | 6E D6 00052 | INCL | CHAR_CNT | : 1482 |
| | 04 | A2 | 01 88 00054 | BISB2 | #1, 4(R2) | : 1483 |
| | 14 | A2 | 01 D0 00058 | MOVL | #1, 20(R2) | : 1484 |
| | | | 00BD 31 0005C | BRW | 14\$ | : 1485 |
| | | 24 | 63 91 0005F | 3\$: CMPB | (R3), #36 | : 1490 |
| | | | 14 12 00062 | BNEQ | 4\$ | |
| | 6E | | 02 C0 00064 | ADDL2 | #2, CHAR_CNT | : 1498 |
| | 04 | A2 | 09 88 00067 | BISB2 | #9, 4(R2) | : 1500 |
| | 14 | A2 | 00000000' EF 3C 0006B | MOVZWL | CURRENCY_DESC, 20(R2) | : 1501 |
| | | 14 | A2 D6 00073 | INCL | 20(R2) | |
| | | | 10 11 00076 | BRB | 5\$ | : 1507 |
| | 2A | | 63 91 00078 | 4\$: CMPB | (R3), #42 | : 1514 |
| | | | 23 12 0007B | BNEQ | 6\$ | |
| | 6E | | 02 C0 0007D | ADDL2 | #2, CHAR_CNT | : 1522 |
| | 04 | A2 | 11 88 00080 | BISB2 | #17, 4(R2) | : 1524 |

| | | | | | | | | | |
|-----------|----|-----------|----|----|-------|-------|--------|--------------------|------|
| 14 | A2 | | 02 | D0 | 00084 | | MOVL | #2, 20(R2) | 1525 |
| | | 4004 | 8F | BB | 00088 | 5\$: | PUSHR | #^M<R2,SP> | 1530 |
| | | | 54 | DD | 0008C | | PUSHL | R4 | |
| | | | 25 | DD | 0008E | | PUSHL | #37 | |
| 0000V | CF | | 04 | FB | 00090 | | CALLS | #4, SPAN_BRACKET | |
| | | | 52 | DD | 00095 | | PUSHL | R2 | 1531 |
| | | 04 | AE | 9F | 00097 | | PUSHAB | CHAR_CNT | |
| | | | 54 | DD | 0009A | | PUSHL | R4 | |
| | | | 02 | DD | 0009C | | PUSHL | #2 | |
| | | | 1A | 11 | 0009E | | BRB | 7\$ | |
| | 2E | | 63 | 91 | 000A0 | 6\$: | CMPB | (R3), #46 | 1536 |
| | | | 17 | 12 | 000A3 | | BNEQ | 8\$ | |
| | 6E | | 02 | C0 | 000A5 | | ADDL2 | #2, CHAR_CNT | 1544 |
| 04 | A2 | 41 | 8F | 88 | 000A8 | | BISB2 | #65, 4(R2) | 1546 |
| 10 | A2 | | 01 | D0 | 000AD | | MOVL | #1, 16(R2) | 1547 |
| | | | 52 | DD | 000B1 | | PUSHL | R2 | 1548 |
| | | 04 | AE | 9F | 000B3 | | PUSHAB | CHAR_CNT | |
| | | | 54 | DD | 000B6 | | PUSHL | R4 | |
| | | | 03 | DD | 000B8 | | PUSHL | #3 | |
| | | | 69 | 11 | 000BA | 7\$: | BRB | 15\$ | |
| 5C | 8F | | 63 | 91 | 000BC | 8\$: | CMPB | (R3), #92 | 1553 |
| | | | 1F | 12 | 000C0 | | BNEQ | 10\$ | |
| | | | 6E | D6 | 000C2 | | INCL | CHAR_CNT | 1555 |
| | | 4004 | 8F | BB | 000C4 | | PUSHR | #^M<R2,SP> | 1556 |
| | | | 54 | DD | 000C8 | | PUSHL | R4 | |
| 0000V | CF | | 03 | FB | 000CA | | CALLS | #3, SPAN_BACKSLASH | |
| | 58 | | 50 | E8 | 000CF | | BLBS | R0, 16\$ | |
| | | 00000000G | 8F | DD | 000D2 | 9\$: | PUSHL | #BAS\$K PRIUSIFOR | 1559 |
| 00000000G | 00 | | 01 | FB | 000D8 | | CALLS | #1, BAS\$\$STOP | |
| | | | 5F | 11 | 000DF | | BRB | 18\$ | 1560 |
| | 21 | | 63 | 91 | 000E1 | 10\$: | CMPB | (R3), #33 | 1568 |
| | | | 0C | 12 | 000E4 | | BNEQ | 11\$ | |
| | | | 6E | D6 | 000E6 | | INCL | CHAR_CNT | 1570 |
| 05 | A2 | | 08 | 88 | 000E8 | | BISB2 | #8, 5(R2) | 1571 |
| 18 | A2 | | 01 | D0 | 000EC | | MOVL | #1, 24(R2) | 1572 |
| | | | 38 | 11 | 000F0 | | BRB | 16\$ | 1573 |
| | 3C | | 63 | 91 | 000F2 | 11\$: | CMPB | (R3), #60 | 1581 |
| | | | 4D | 12 | 000F5 | | BNEQ | 19\$ | |
| | 50 | 01 | A3 | 9A | 000F7 | | MOVZBL | 1(R3), R0 | 1585 |
| | 25 | | 50 | 91 | 000FB | | CMPB | R0, #37 | 1587 |
| | | | 0B | 12 | 000FE | | BNEQ | 12\$ | |
| | 6E | | 03 | C0 | 00100 | | ADDL2 | #3, CHAR_CNT | 1589 |
| 04 | A2 | 1001 | 8F | A8 | 00103 | | BISW2 | #4097, 4(R2) | 1591 |
| | | | 0E | 11 | 00109 | | BRB | 13\$ | 1592 |
| | 30 | | 50 | 91 | 0010B | 12\$: | CMPB | R0, #48 | 1598 |
| | | | 21 | 12 | 0010E | | BNEQ | 17\$ | |
| | 6E | | 03 | C0 | 00110 | | ADDL2 | #3, CHAR_CNT | 1600 |
| 04 | A2 | 2001 | 8F | A8 | 00113 | | BISW2 | #8193, 4(R2) | 1602 |
| | | 14 | A2 | D6 | 00119 | 13\$: | INCL | 20(R2) | 1603 |
| | | | 52 | DD | 0011C | 14\$: | PUSHL | R2 | 1604 |
| | | 04 | AE | 9F | 0011E | | PUSHAB | CHAR_CNT | |
| | | | 54 | DD | 00121 | | PUSHL | R4 | |
| | | | 01 | DD | 00123 | | PUSHL | #1 | |
| 0000V | CF | | 04 | FB | 00125 | 15\$: | CALLS | #4, SPAN_NUMERIC | 1606 |
| 0000V | CF | | 52 | DD | 0012A | 16\$: | PUSHL | R2 | 1606 |
| | | | 01 | FB | 0012C | | CALLS | #1, OUTPUT_ARG | |
| | | | 5E | DD | 00131 | 17\$: | PUSHL | SP | 1637 |

BASS\$FOR_INT
2-013

BASS\$FORMAT_INT - Basic format interpreter
DECODE_PROCESS_FORMAT - main driving routine

N 6
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 24
(4)

| | | | | | | | | | |
|----|----|----|----|----|-------|-------|-------|-------------------------|--|
| | | | 62 | DD | 00133 | | PUSHL | (R2) | |
| | | | 54 | DD | 00135 | | PUSHL | R4 | |
| | | | 03 | FB | 00137 | | CALLS | #3, SPAN_CONSTANT | |
| 20 | A2 | | 6E | C1 | 0013A | | ADDL3 | CHAR_CNT, 4(R4), 32(R2) | |
| | | 04 | 01 | D0 | 00140 | 18\$: | MOVL | #1, R0 | |
| | | | | | 04 | | RET | | |
| | | | 50 | D4 | 00144 | 19\$: | CLRL | R0 | |
| | | | | | 04 | | RET | | |

.....
..... 1644
..... 1646
.....
..... 1648
.....

; Routine Size: 327 bytes, Routine Base: _BAS\$CODE + 01E6

```

: 947      1649  1 %SBTTL 'SPAN_CONSTANT - find start of format sequence'
: 948      1650  1 ROUTINE SPAN_CONSTANT (
: 949      1651  1
: 950      1652  1     P_FORMAT_STR,  ! format string
: 951      1653  1     P_RETURN_STR,  ! return string
: 952      1654  1     P_CHAR_CNT) = ! character position within format string
: 953      1655  1
: 954      1656  2 BEGIN
: 955      1657  2
: 956      1658  2 !++
: 957      1659  2
: 958      1660  2 FUNCTIONAL DESCRIPTION:
: 959      1661  2
: 960      1662  2     This routine searches FORMAT_STR for the start of a valid
: 961      1663  2     formatting sequence beginning at the CHAR_CNT + 1st character
: 962      1664  2     position in FORMAT_STR. CHAR_CNT is set to the character position
: 963      1665  2     just before the formatting sequence. Thus CHAR_CNT returned
: 964      1666  2     unchanged indicates that no constant string was found, i.e.,
: 965      1667  2     CHAR_CNT currently points to the character position just before
: 966      1668  2     the start of a format sequence. The constant string is copied
: 967      1669  2     onto the return string. The routine returns 0 if no
: 968      1670  2     formatting sequence is located and 1 otherwise.
: 969      1671  2
: 970      1672  2     This routine will be called initially to determine the end of
: 971      1673  2     any leading constant in the format string. This routine
: 972      1674  2     will be called after any formatting has taken place to determine
: 973      1675  2     the end of any trailing constant in the format string.
: 974      1676  2
: 975      1677  2 CALLING SEQUENCE:
: 976      1678  2
: 977      1679  2     STATUS.wlc.v = SPAN_CONSTANT (P_FORMAT_STR.rt.dx, P_RETURN_STR.mt.dd,
: 978      1680  2     P_CHAR_CNT.ml.r)
: 979      1681  2
: 980      1682  2 FORMAL PARAMETERS:
: 981      1683  2
: 982      1684  2     P_FORMAT_STR - address of string descriptor for format string
: 983      1685  2     P_RETURN_STR - address of string descriptor for return string
: 984      1686  2     P_CHAR_CNT   - address of longword for character position within
: 985      1687  2     format string; this parameter is passed as the
: 986      1688  2     position 1 before the point at which the format
: 987      1689  2     sequence should start; this parameter is passed
: 988      1690  2     as the position 1 before the starting character
: 989      1691  2     position of the format sequence
: 990      1692  2
: 991      1693  2 IMPLICIT INPUTS:
: 992      1694  2
: 993      1695  2     NONE
: 994      1696  2
: 995      1697  2 IMPLICIT OUTPUTS:
: 996      1698  2
: 997      1699  2     NONE
: 998      1700  2
: 999      1701  2 COMPLETION CODES:
: 1000     1702  2
: 1001     1703  2     1 - routine completed successfully (format sequence
: 1002     1704  2     located)
: 1003     1705  2     0 - no format sequence located

```

```

: 1004      1706  2 |
: 1005      1707  2 | SIDE EFFECTS:
: 1006      1708  2 |
: 1007      1709  2 |     NONE
: 1008      1710  2 |
: 1009      1711  2 | EXAMPLES:
: 1010      1712  2 |
: 1011      1713  2 |     FORMAT_STR      CHAR_CNT (in)  CHAR_CNT (out)  COMPLETION CODE
: 1012      1714  2 |     -----
: 1013      1715  2 |     abc$abc          0             0             0
: 1014      1716  2 |     abc$$abc         0             3             1
: 1015      1717  2 |     abc$abc**abc     0             7             1
: 1016      1718  2 |     abc'cccdef'cc   7             10            1
: 1017      1719  2 |     abc'cccdef*     7             7             0
: 1018      1720  2 |     abc'cccdef**    7             10            1
: 1019      1721  2 |     abc'ccc         7             7             0
: 1020      1722  2 |
: 1021      1723  2 | --
: 1022      1724  2 |
: 1023      1725  2 | BIND
: 1024      1726  2 |     FORMAT_STR = (.P_FORMAT_STR) : BLOCK [, BYTE],
: 1025      1727  2 |     RETURN_STR = (.P_RETURN_STR) : BLOCK [, BYTE],
: 1026      1728  2 |     CHAR_CNT = (.P_CHAR_CNT);
: 1027      1729  2 |
: 1028      1730  2 | LOCAL
: 1029      1731  2 |     TMP_STR : BLOCK [8, BYTE],      | temporary string descriptor
: 1030      1732  2 |     FORMAT_STR_LEN : UNSIGNED WORD, | temporary for format length in SCANC
: 1031      1733  2 |     SCAN_MASK : BYTE,              | temporary for mask in SCANC
: 1032      1734  2 |     START_CNS_CNT,                 | initial offset that could possibly
: 1033      1735  2 |                                     | begin a constant in the format
: 1034      1736  2 |                                     | string
: 1035      1737  2 |     CHAR_PTR;                       | pointer to possible start point
: 1036      1738  2 |                                     | for formatting sequence
: 1037      1739  2 |
: 1038      1740  2 | +
: 1039      1741  2 | | START_CNS_CNT should be the initial offset that could possible
: 1040      1742  2 | | begin a constant string.
: 1041      1743  2 | -
: 1042      1744  2 |
: 1043      1745  2 | START_CNS_CNT = .CHAR_CNT + 1;
: 1044      1746  2 |
: 1045      1747  2 | +
: 1046      1748  2 | | Use SCANC to find the beginning of a formatting sequence.
: 1047      1749  2 | | SCANC will be used to find the first 'possible' formatting character.
: 1048      1750  2 | | Since SCANC can only stop on single character sequences, and we only
: 1049      1751  2 | | want to stop on multiple character sequences in some cases
: 1050      1752  2 | | ('.', '#', '**' and '$$),
: 1051      1753  2 | | SCANC may have to be used repeatedly to skip over false alarms such
: 1052      1754  2 | | as '.R', '$2' or '*b'.
: 1053      1755  2 | -
: 1054      1756  2 |
: 1055      1757  2 | WHILE 1 DO                          | This loop scans until the end
: 1056      1758  2 |                                     | of a leading or trailing constant
: 1057      1759  2 |                                     | is found
: 1058      1760  2 |     BEGIN
: 1059      1761  2 |
: 1060      1762  2 |     FORMAT_STR_LEN = .FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT;

```

```

1061 1763 3      SCAN_MASK = MASK_FORMAT CHARS;
1062 1764 3      CHAR_PTR = SCANC (FORMAT_STR_LEN,
1063 1765 3      .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT,
1064 1766 3      CLASS_TABLE, SCAN_MASK);
1065 1767 3      IF .CHAR_PTR EQLU 0
1066 1768 3      THEN
1067 1769 4          BEGIN
1068 1770 4
1069 1771 4      !+
1070 1772 4      ! If CHAR_CNT is not 0, then we are trying to span a trailing constant.
1071 1773 4      ! If no formatting sequence is found, we should treat everything from
1072 1774 4      ! START_CNS_CNT to the end of the string as a trailing constant.
1073 1775 4      !-
1074 1776 4
1075 1777 4          IF .CHAR_CNT NEQU 0
1076 1778 4          THEN
1077 1779 4
1078 1780 5              BEGIN
1079 1781 5              IF .FORMAT_STR [DSC$W_LENGTH] - .START_CNS_CNT GEQ 0
1080 1782 5              THEN
1081 1783 6                  BEGIN
1082 1784 6                  TMP_STR [DSC$W_LENGTH] = 0;
1083 1785 6                  TMP_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
1084 1786 6                  TMP_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
1085 1787 6                  TMP_STR [DSC$A_POINTER] = 0;
1086 1788 6                  STR$RIGHT (TMP_STR, FORMAT_STR, START_CNS_CNT);
1087 1789 6                  STR$CONCAT (RETURN_STR, RETURN_STR, TMP_STR);
1088 1790 6                  STR$FREE1_DX (TMP_STR);
1089 1791 6                  CHAR_CNT = .FORMAT_STR [DSC$W_LENGTH];
1090 1792 6                  END
1091 1793 4              END;
1092 1794 4
1093 1795 4          RETURN 0;          ! return since end of format string
1094 1796 4          ! was found
1095 1797 4          END
1096 1798 4
1097 1799 3      ELSE
1098 1800 4          BEGIN
1099 1801 4          BIND CHARS = (.CHAR_PTR) : VECTOR [, BYTE];
1100 1802 4          CHAR_CNT = .CHAR_PTR - .FORMAT_STR [DSC$A_POINTER];
1101 1803 4
1102 1804 4      !+
1103 1805 4      ! Check special cases ('.#', '$$', '**', '<CD>', '<X>', and '<0>').
1104 1806 4      !-
1105 1807 4
1106 1808 4          IF .CHARS [0] EQLU CHAR_PERIOD
1107 1809 4          THEN
1108 1810 5              BEGIN
1109 1811 5              IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] - 1
1110 1812 5              THEN
1111 1813 6                  BEGIN
1112 1814 6                  IF .CHARS [1] EQLU CHAR_POUND
1113 1815 6                  THEN
1114 1816 6                      EXITLOOP
1115 1817 6
1116 1818 6                  ELSE
1117 1819 6

```

```

: 1118      1820  6  !+
: 1119      1821  6  ! The format character was a false alarm so we should return to top of loop
: 1120      1822  6  ! and get the possible format character into the constant.
: 1121      1823  6  !-
: 1122      1824  6
: 1123      1825  6
: 1124      1826  6          CHAR_CNT = .CHAR_CNT + 1;
: 1125      1827  6          ! this gets the case "abc.+"
: 1126      1828  6          END
: 1127      1829  6          ELSE
: 1128      1830  5
: 1129      1831  5  !+
: 1130      1832  5  ! The format character was a false alarm so we should return to top of loop
: 1131      1833  5  ! and get the possible format character into the constant.
: 1132      1834  5  !-
: 1133      1835  5
: 1134      1836  5          CHAR_CNT = .CHAR_CNT + 1;
: 1135      1837  5          ! this gets the case "abc."
: 1136      1838  5          END
: 1137      1839  5
: 1138      1840  4  ELSE IF .CHARS [0] EQLU CHAR_DOLLAR
: 1139      1841  4  THEN
: 1140      1842  5      BEGIN
: 1141      1843  5      IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] - 1
: 1142      1844  5      THEN
: 1143      1845  6          BEGIN
: 1144      1846  6          IF .CHARS [1] EQLU CHAR_DOLLAR
: 1145      1847  6          THEN
: 1146      1848  6              EXITLOOP
: 1147      1849  6
: 1148      1850  6          ELSE
: 1149      1851  6
: 1150      1852  6  !+
: 1151      1853  6  ! The format character was a false alarm so we should return to top of loop
: 1152      1854  6  ! and get the possible format character into the constant.
: 1153      1855  6  !-
: 1154      1856  6
: 1155      1857  6          CHAR_CNT = .CHAR_CNT + 1;
: 1156      1858  6          ! this gets the case "abc$+"
: 1157      1859  6          END
: 1158      1860  6          ELSE
: 1159      1861  5
: 1160      1862  5  !+
: 1161      1863  5  ! The format character was a false alarm so we should return to top of loop
: 1162      1864  5  ! and get the possible format character into the constant.
: 1163      1865  5  !-
: 1164      1866  5
: 1165      1867  5
: 1166      1868  5          CHAR_CNT = .CHAR_CNT + 1;
: 1167      1869  5          ! this gets the case "abc$"
: 1168      1870  5          END
: 1169      1871  5
: 1170      1872  4  ELSE IF .CHARS [0] EQLU CHAR_STAR
: 1171      1873  4  THEN
: 1172      1874  5      BEGIN
: 1173      1875  5      IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] - 1
: 1174      1876  5      THEN

```

```

: 1175      1877 6
: 1176      1878 6
: 1177      1879 6
: 1178      1880 6
: 1179      1881 6
: 1180      1882 6
: 1181      1883 6
: 1182      1884 6
: 1183      1885 6
: 1184      1886 6
: 1185      1887 6
: 1186      1888 6
: 1187      1889 6
: 1188      1890 6
: 1189      1891 6
: 1190      1892 6
: 1191      1893 5
: 1192      1894 5
: 1193      1895 5
: 1194      1896 5
: 1195      1897 5
: 1196      1898 5
: 1197      1899 5
: 1198      1900 5
: 1199      1901 5
: 1200      1902 5
: 1201      1903 5
: 1202      1904 4
: 1203      1905 4
: 1204      1906 5
: 1205      1907 5
: 1206      1908 5
: 1207      1909 6
: 1208      1910 6
: 1209      1911 6
: 1210      1912 7
: 1211      1913 7
: 1212      1914 7
: 1213      1915 6
: 1214      1916 6
: 1215      1917 6
: 1216      1918 6
: 1217      1919 6
: 1218      1920 6
: 1219      1921 6
: 1220      1922 6
: 1221      1923 7
: 1222      1924 6
: 1223      1925 6
: 1224      1926 7
: 1225      1927 7
: 1226      1928 7
: 1227      1929 7
: 1228      1930 7
: 1229      1931 7
: 1230      1932 7
: 1231      1933 7

```

```

      BEGIN
      IF .CHARS [1] EQLU CHAR_STAR
      THEN
          EXITLOOP
      ELSE
          !+ The format character was a false alarm so we should return to top of loop
          ! and get the possible format character into the constant.
          CHAR_CNT = .CHAR_CNT + 1;
          ! this gets the case "abc*+"
      END
      ELSE
          !+ The format character was a false alarm so we should return to top of loop
          ! and get the possible format character into the constant.
          CHAR_CNT = .CHAR_CNT + 1;
          ! this gets the case "abc*"
      END
      ELSE IF .CHARS [0] EQLU CHAR_UNDERSCORE
      THEN
          BEGIN
          IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] -1
          THEN
              BEGIN
              LOCAL
              UNDER_MASK;
              UNDER_MASK = (MASK_FORMAT_CHARS +
                          MASK_L + MASK_C +
                          MASK_R + MASK_E +
                          MASK_SPECIAL);
              !+ Underscore must be the first character in the
              ! constant. Any other
              ! constant characters should be appended to the
              ! return string before
              ! we try to process underscore.
              IF .CHAR_PTR - (.FORMAT_STR [DSC$A_POINTER] +
                          .START_CNS_CNT) GEQ 0
              THEN
                  BEGIN
                  !+ Underscore is in the middle
                  ! of the constant. Include
                  ! up to the _ in the return
                  ! string now, and set the
                  ! start ptr to the _ for next
                  ! time thru SPAN_CONSTANT.

```

```

: 1232 1934 7
: 1233 1935 7
: 1234 1936 7
: 1235 1937 7
: 1236 1938 7
: 1237 1939 7
: 1238 1940 7
: 1239 1941 7
: 1240 1942 7
: 1241 1943 7
: 1242 1944 7
: 1243 1945 7
: 1244 1946 7
: 1245 1947 6
: 1246 1948 7
: 1247 1949 7
: 1248 1950 7
: 1249 1951 7
: 1250 1952 7
: 1251 1953 7
: 1252 1954 7
: 1253 1955 7
: 1254 1956 7
: 1255 1957 7
: 1256 1958 7
: 1257 1959 7
: 1258 1960 7
: 1259 1961 7
: 1260 1962 7
: 1261 1963 7
: 1262 1964 7
: 1263 1965 7
: 1264 1966 7
: 1265 1967 7
: 1266 1968 7
: 1267 1969 7
: 1268 1970 7
: 1269 1971 7
: 1270 1972 7
: 1271 1973 7
: 1272 1974 7
: 1273 1975 7
: 1274 1976 7
: 1275 1977 7
: 1276 1978 7
: 1277 1979 7
: 1278 1980 7
: 1279 1981 7
: 1280 1982 8
: 1281 1983 8
: 1282 198 8
: 1283 1985 9
: 1284 1986 9
: 1285 1987 9
: 1286 1988 9
: 1287 1989 9
: 1288 1990 9

```

```

!-
LOCAL
  CNS_LENGTH;
  CNS_LENGTH = .CHAR_CNT - .START_CNS_CNT + 1;
  TMP_STR [DSC$W_LENGTH] = 0;
  TMP_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
  TMP_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
  TMP_STR [DSC$A_POINTER] = 0;
  STR$LEN_EXTR (TMP_STR, FORMAT_STR, START_CNS_CNT, CNS_LENGTH);
  STR$CONCAT (RETURN_STR, RETURN_STR, TMP_STR);
  STR$FREE1_DX (TMP_STR);
  START_CNS_CNT = .CHAR_CNT + 1; ! constant starts past underscore
  END ! the case 'abc_###'
ELSE
  BEGIN ! start of constant
  +
  Search for the formatting character which
  should follow an underscore.
  Underscore has no effect if no format
  character follows.
  Note we don't EXITLOOP after processing the
  underscore and the char that
  follows it, as there may be more constant
  expression after that.
  -
  FORMAT_STR_LEN = 1;
  CHAR_PTR = SCANC (FORMAT_STR_LEN,
    .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT + 1,
    CLASS_TABLE, UNDER_MASK);
  IF .CHAR_PTR EQ 0
  THEN
  +
  No format character follows the underscore,
  so the underscore should become
  part of the constant.
  -
  CHAR_CNT = .CHAR_CNT + 1
  ! this gets the case '_abc'
  ELSE
  +
  A valid text format character follows the
  underscore. This character should
  be included in the constant. There may be
  more constant after this single
  character - SCANC again, and if there is
  another format character we know
  we are at the end of the constant.
  -
  BEGIN
  IF .CHAR_CNT EQLU .FORMAT_STR [DSC$W_LENGTH] - 2
  THEN
  BEGIN
  +
  Nothing follows the underscore and its
  format character. End of constant.
  -
  IF .CHAR_CNT EQL 0

```

```

: 1289 1991 9
: 1290 1992 9
: 1291 1993 9
: 1292 1994 9
: 1293 1995 9
: 1294 1996 9
: 1295 1997 9
: 1296 1998 9
: 1297 1999 9
: 1298 2000 9
: 1299 2001 8
: 1300 2002 9
: 1301 2003 9
: 1302 2004 9
: 1303 2005 9
: 1304 2006 9
: 1305 2007 9
: 1306 2008 9
: 1307 2009 9
: 1308 2010 9
: 1309 2011 9
: 1310 2012 9
: 1311 2013 9
: 1312 2014 10
: 1313 2015 10
: 1314 2016 10
: 1315 2017 10
: 1316 2018 10
: 1317 2019 10
: 1318 2020 10
: 1319 2021 10
: 1320 2022 10
: 1321 2023 10
: 1322 2024 10
: 1323 2025 10
: 1324 2026 11
: 1325 2027 11
: 1326 2028 11
: 1327 2029 11
: 1328 2030 10
: 1329 2031 10
: 1330 2032 10
: 1331 2033 10
: 1332 2034 9
: 1333 2035 10
: 1334 2036 10
: 1335 2037 10
: 1336 2038 10
: 1337 2039 10
: 1338 2040 10
: 1339 2041 10
: 1340 2042 10
: 1341 2043 10
: 1342 2044 10
: 1343 2045 10
: 1344 2046 10
: 1345 2047 10

```

```

THEN
  RETURN 0; ! no format chars found
CHAR_CNT = CHAR_CNT + 2;
IF .CHARS [1] NEQU CHAR_DOLLAR AND
   .CHARS [1] NEQU CHAR_STAR AND
   .CHARS [1] NEQU CHAR_LF_ANGLE
THEN
  START_CNS_CNT = .START_CNS_CNT + 1;
EXITLOOP
END ! case '# ' (end of string)
ELSE
BEGIN
+
  Characters which are part of a '<x>'
  sequence may not normally be considered
  format characters.
-
  UNDER_MASK = .UNDER_MASK + MASK_LF_ANGLE;
  CHAR_PTR = SCANC (FORMAT_STR_LEN,
    .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT + 2,
    CLASS_TABLE, UNDER_MASK);
  IF .CHAR_PTR EQL 0
  THEN
  BEGIN
  +
    More constant follows the underscore and
    its format character. If multi-char
    format, then the char after the
    underscore is not a 'format' char
    after all.
  -
    IF .CHARS [1] NEQU CHAR_DOLLAR AND
     .CHARS [1] NEQU CHAR_STAR AND
     .CHARS [1] NEQU CHAR_LF_ANGLE
    THEN
    BEGIN
      CHAR_CNT = .CHAR_CNT + 2;
      START_CNS_CNT = .START_CNS_CNT + 1;
    END ! case '#abc'
  ELSE
    CHAR_CNT = .CHAR_CNT + 1;
  END ! case '<abc'
  END
ELSE
BEGIN
+
  More format characters follow the underscore
  and its format character.
  Check for a multi-char format sequence,
  such as '$$'. Include all chars in
  the constant if necessary.
-
  BIND
  FORMAT_CHAR = (.CHAR_PTR - 1) : VECTOR [,BYTE];
  SELECTONEU .FORMAT_CHAR [0] OF
  SET
  [CHAR_DOLLAR] :

```

```

: 1346 2048 10
: 1347 2049 10
: 1348 2050 11
: 1349 2051 11
: 1350 2052 11
: 1351 2053 11
: 1352 2054 10
: 1353 2055 10
: 1354 2056 10
: 1355 2057 10
: 1356 2058 10
: 1357 2059 10
: 1358 2060 11
: 1359 2061 11
: 1360 2062 11
: 1361 2063 11
: 1362 2064 10
: 1363 2065 10
: 1364 2066 10
: 1365 2067 10
: 1366 2068 10
: 1367 2069 10
: 1368 2070 11
: 1369 2071 11
: 1370 2072 11
: 1371 2073 11
: 1372 2074 11
: 1373 2075 12
: 1374 2076 12
: 1375 2077 12
: 1376 2078 11
: 1377 2079 11
: 1378 2080 10
: 1379 2081 10
: 1380 2082 10
: 1381 2083 10
: 1382 2084 11
: 1383 2085 11
: 1384 2086 11
: 1385 2087 12
: 1386 2088 12
: 1387 2089 12
: 1388 2090 12
: 1389 2091 12
: 1390 2092 12
: 1391 2093 13
: 1392 2094 13
: 1393 2095 12
: 1394 2096 13
: 1395 2097 13
: 1396 2098 13
: 1397 2099 12
: 1398 2100 13
: 1399 2101 13
: 1400 2102 13
: 1401 2103 13
: 1402 2104 12

```

```

IF .FORMAT_CHAR [1] EQLU CHAR_DOLLAR
THEN
  BEGIN
    CHAR_CNT = .CHAR_CNT + 3;
    START_CNS_CNT = .START_CNS_CNT + 1;
  END ! case '$_$'
ELSE
  CHAR_CNT = .CHAR_CNT + 1;
  ! case '$_#'
[CHAR_STAR] :
IF .FORMAT_CHAR [1] EQLU CHAR_STAR
THEN
  BEGIN
    CHAR_CNT = .CHAR_CNT + 3;
    START_CNS_CNT = .START_CNS_CNT + 1;
  END ! case '$_**'
ELSE
  CHAR_CNT = .CHAR_CNT + 1;
  ! case '$_*#'
[CHAR_CARAT] :
IF .CHAR_CNT LEQU .FORMAT_STR [DSC$W_LENGTH] - 5
THEN
  BEGIN
    IF .FORMAT_CHAR [1] EQLU CHAR_CARAT AND
        .FORMAT_CHAR [2] EQLU CHAR_CARAT AND
        .FORMAT_CHAR [3] EQLU CHAR_CARAT
    THEN
      BEGIN
        CHAR_CNT = .CHAR_CNT + 5;
        START_CNS_CNT = .START_CNS_CNT + 1;
      END ! case '$_^'^'^'^'^'
    ELSE
      CHAR_CNT = .CHAR_CNT + 2;
      ! case '$_^#^'
[CHAR_LF_ANGLE] :
BEGIN
IF .CHAR_CNT LEQU .FORMAT_STR [DSC$W_LENGTH] - 4
THEN
  BEGIN
    LOCAL
      TMP_CHAR_CNT,
      TMP_PARAM_BLK : BLOCK [K_INTER_STO_LEN, BYTE];
    ! don't alter real param_blk
    TMP_CHAR_CNT = .CHAR_CNT + 1;
    IF (SPAN_BRACKET (CHAR_PERCENT,
      FORMAT_STR, TMP_PARAM_BLK,
      TMP_CHAR_CNT)) OR
      (SPAN_BRACKET (CHAR_ZERO,
      FORMAT_STR, TMP_PARAM_BLK,
      TMP_CHAR_CNT))
    THEN
      ! cases '$_<%' , '$_<0%'
      BEGIN
        CHAR_CNT = .CHAR_CNT + 4;
        START_CNS_CNT = .START_CNS_CNT + 1;
      END
    ELSE

```

```

: 1403      2105 13
: 1404      2106 13
: 1405      2107 13
: 1406      2108 13
: 1407      2109 13
: 1408      2110 13
: 1409      2111 12
: 1410      2112 13
: 1411      2113 13
: 1412      2114 13
: 1413      2115 13
: 1414      2116 12
: 1415      2117 12
: 1416      2118 12
: 1417      2119 12
: 1418      2120 11
: 1419      2121 11
: 1420      2122 10
: 1421      2123 10
: 1422      2124 10
: 1423      2125 10
: 1424      2126 10
: 1425      2127 10
: 1426      2128 10
: 1427      2129 10
: 1428      2130 10
: 1429      2131 10
: 1430      2132 11
: 1431      2133 11
: 1432      2134 11
: 1433      2135 11
: 1434      2136 10
: 1435      2137 10
: 1436      2138 10
: 1437      2139 9
: 1438      2140 9
: 1439      2141 9
: 1440      2142 8
: 1441      2143 7
: 1442      2144 6
: 1443      2145 6
: 1444      2146 6
: 1445      2147 6
: 1446      2148 5
: 1447      2149 6
: 1448      2150 6
: 1449      2151 6
: 1450      2152 6
: 1451      2153 6
: 1452      2154 6
: 1453      2155 5
: 1454      2156 5
: 1455      2157 5
: 1456      2158 4
: 1457      2159 4
: 1458      2160 5
: 1459      2161 5

```

```

IF (SPAN BRACKET (CHAR C,
                  FORMAT_STR, TMP_PARAM_BLK,
                  TMP_CHAR_CNT)) OR
   (SPAN BRACKET (CHAR_LOWER_C,
                  FORMAT_STR, TMP_PARAM_BLK,
                  TMP_CHAR_CNT))
THEN
    ! cases '_<CD>', '_<cd>'
    BEGIN
    CHAR_CNT = .CHAR_CNT + 5;
    START_CNS_CNT = .START_CNS_CNT + 1;
    END
ELSE
    CHAR_CNT = .CHAR_CNT + 3;
    ! case '_<%abc'
    END
ELSE
    CHAR_CNT = .CHAR_CNT + 3; ! case '_<%
    END;
    ! end of bracket formats

[OTHERWISE] :
    +
    ! Underscore followed by a
    ! singly valid format char.
    ! Any format chars which
    ! follow the 1st format char
    ! are immaterial.
    -
    BEGIN
    CHAR_CNT = .CHAR_CNT + 2;
    START_CNS_CNT = .START_CNS_CNT + 1;
    ! case '_#abc'
    END;
    TES;
    END; ! more format chars after
    ! underscore & its format char

    END
    END
    END
    END
    END
    +
    ! underscore followed by nothing. treat as constant.
    -
    ELSE
    BEGIN
    +
    -
    CHAR_CNT = .CHAR_CNT + 1;
    EXIT[COOP];
    END;
    END

ELSE IF .CHARS [0] EQLU CHAR_LF_ANGLE
THEN
    BEGIN
    IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] - 2

```

```

: 1460      2162  5      THEN
: 1461      2163  6      BEGIN
: 1462      2164  6      SELECTONEU .CHARS [1] OF
: 1463      2165  6      SET
: 1464      2166  6      [CHAR PERCENT]:
: 1465      2167  6      IF .CHARS [2] EQLU CHAR_RT_ANGLE
: 1466      2168  6      THEN ! found a format character
: 1467      2169  6      EXITLOOP
: 1468      2170  6      ELSE ! false alarm - include in constant
: 1469      2171  6      CHAR_CNT = .CHAR_CNT + 2;
: 1470      2172  6      ! this gets the case '<Zabc'
: 1471      2173  6      [CHAR ZERO]:
: 1472      2174  6      IF .CHARS [2] EQLU CHAR_RT_ANGLE
: 1473      2175  6      THEN ! found a format character
: 1474      2176  6      EXITLOOP
: 1475      2177  6      ELSE ! false alarm - include in constant
: 1476      2178  6      CHAR_CNT = .CHAR_CNT + 2;
: 1477      2179  6      ! this gets the case '<0abc'
: 1478      2180  6      [CHAR C]:
: 1479      2181  6      IF .CHARS [2] EQLU CHAR_D
: 1480      2182  6      THEN
: 1481      2183  7      BEGIN
: 1482      2184  7      IF .CHAR_CNT LSSU .FORMAT_STR [DSC$W_LENGTH] - 3
: 1483      2185  7      THEN
: 1484      2186  8      BEGIN
: 1485      2187  8      IF .CHARS [3] EQLU CHAR_RT_ANGLE
: 1486      2188  8      THEN
: 1487      2189  9      BEGIN
: 1488      2190  9      IF .CHAR_CNT GEQU .FORMAT_STR [DSC$W_LENGTH] - 4
: 1489      2191  9      IF .CHARS[NEQ .FORMAT_STR [DSC$A_POINTER]
: 1490      2192  9      THEN
: 1491      2193  9      EXITLOOP ! found a format char (at end of string)
: 1492      2194  9      ELSE
: 1493      2195  9      CHAR_CNT = .CHAR_CNT + 4
: 1494      2196  9      ! this gets the case where <CD> is in the beginning of
: 1495      2197  9      ! the format string and should be treated as a constant.
: 1496      2198  9      ! we treat it as a constant in order to be consistent w/ BP2
: 1497      2199  9      END
: 1498      2200  8      ELSE
: 1499      2201  8      CHAR_CNT = .CHAR_CNT + 3
: 1500      2202  8      ! this gets the case '<CDabc'
: 1501      2203  8      END
: 1502      2204  7      ELSE
: 1503      2205  7      CHAR_CNT = .CHAR_CNT + 3
: 1504      2206  7      ! this gets the case '<CDabc' at end of string
: 1505      2207  7      END
: 1506      2208  6      ELSE
: 1507      2209  6      CHAR_CNT = .CHAR_CNT + 2;
: 1508      2210  6      ! this gets the case '<Cabc'
: 1509      2211  6      [OTHERWISE]:
: 1510      2212  6      CHAR_CNT = .CHAR_CNT + 1;
: 1511      2213  6      ! this gets the case '<abc'
: 1512      2214  6      TES;
: 1513      2215  6      END
: 1514      2216  5      ELSE
: 1515      2217  5      CHAR_CNT = .CHAR_CNT + 1;
: 1516      2218  5      ! this gets the case '<a'

```

```

: 1517      2219 5          END
: 1518      2220 4          ELSE
: 1519      2221 4          EXITLOOP
: 1520      2222 4          END
: 1521      2223 2          END;
: 1522      2224 2          ! end of constant finding loop
: 1523      2225 2
: 1524      2226 2          +
: 1525      2227 2          If we get here we have moved CHAR_CNT to the character before
: 1526      2228 2          the start of a format string.
: 1527      2229 2          Concatenate constant string from FORMAT_STR onto RETURN_STR.
: 1528      2230 2
: 1529      2231 2          Note that code similar to this is executed at the beginning of
: 1530      2232 2          this module if a trailing constant is detected that extends to the
: 1531      2233 2          end of the format string.
: 1532      2234 2          -
: 1533      2235 2
: 1534      2236 2          IF .CHAR_CNT - .START_CNS_CNT GEQ 0
: 1535      2237 2          THEN
: 1536      2238 2              BEGIN
: 1537      2239 2              LOCAL CNS_LENGTH;
: 1538      2240 2              CNS_LENGTH = .CHAR_CNT - .START_CNS_CNT + 1;
: 1539      2241 2              TMP_STR [DSC$W_LENGTH] = 0;
: 1540      2242 2              TMP_STR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 1541      2243 2              TMP_STR [DSC$B_CLASS] = DSC$K_CLASS_D;
: 1542      2244 2              TMP_STR [DSC$A_POINTER] = 0;
: 1543      2245 2              STR$LEN_EXTR (TMP_STR, FORMAT_STR, START_CNS_CNT, CNS_LENGTH);
: 1544      2246 2              STR$CONCAT (RETURN_STR, RETURN_STR, TMP_STR);
: 1545      2247 2              STR$FREE1_DX (TMP_STR)
: 1546      2248 2          END;
: 1547      2249 2
: 1548      2250 2          RETURN 1
: 1549      2251 2
: 1550      2252 1          END;
! End of routine SPAN_CONSTANT

```

| | | | | OFFC 0000 SPAN_CONSTANT: | | | | | | | |
|----|------|----|---------|--------------------------|----|----|-------|------------|--------------------------------------|--|--------|
| | | | 5E | BC | AE | 9E | 00002 | .WORD | Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 | : 1650 | |
| | | | 59 | 04 | AC | D0 | 00006 | MOVAB | -68(SP), SP | : 1726 | |
| | | | 5B | 08 | AC | D0 | 0000A | MOVL | P_FORMAT_STR, R9 | : 1727 | |
| | | | 56 | 0C | AC | D0 | 0000E | MOVL | P_RETURN_STR, R11 | : 1728 | |
| | 14 | AE | 66 | | 01 | C1 | 00012 | 1\$: ADDL3 | #T, (R6), START_CNS_CNT | : 1745 | |
| | 04 | AE | 69 | | 66 | A3 | 00017 | 2\$: SUBW3 | (R6), (R9), FORMAT_STR_LEN | : 1762 | |
| | | | 6E | | 01 | 90 | 0001C | MOVB | #1, SCAN_MASK | : 1763 | |
| | | | 58 | | 04 | A9 | 0001F | MOVL | 4(R9), R8 | : 1765 | |
| 6E | FCAC | CF | 00 B648 | | 04 | AE | 2A | 00023 | SCANC | FORMAT_STR_LEN, @0(R6)[R8], CLASS_TABLE, - | : 1764 |
| | | | | | | | | | SCAN_MASK | | |
| | | | | | | 02 | 12 | 0002D | BNEQ | 3\$ | |
| | | | | | | 51 | D4 | 0002F | CLRL | R1 | |
| | | | 54 | | | 51 | D0 | 00031 | 3\$: MOVL | R1, CHAR_PTR | |
| | | | | | | 44 | 12 | 00034 | BNEQ | 5\$ | : 1767 |
| | | | | | | 66 | D5 | 00036 | TSTL | (R6) | : 1777 |
| | | | | | | 3D | 13 | 00038 | BEQL | 4\$ | |

| | | | | | | | | | | | | |
|----|------|-----------|----|----|----------|------|----|-------|--------|---------------------------------------|---------------------|------|
| | | | 08 | AE | 01 | A0 | 9E | 000EE | MOVAB | 1(R0), CNS_LENGTH | | |
| | | | 3C | AE | 020E0000 | 8F | D0 | 000F3 | MOVL | #34471936, -TMP_STR | | 1938 |
| | | | | | 40 | AE | D4 | 000FB | CLRL | TMP_STR+4 | | 1941 |
| | | | | | 08 | AE | 9F | 000FE | PUSHAB | CNS_LENGTH | | 1942 |
| | | | | | 18 | AE | 9F | 00101 | PUSHAB | START_CNS_CNT | | |
| | | | | | | 59 | DD | 00104 | PUSHL | R9 | | |
| | | | | | 48 | AE | 9F | 00106 | PUSHAB | TMP_STR | | |
| | | 00000000G | | 00 | | 04 | FB | 00109 | CALLS | #4, -STR\$LEN_EXTR | | |
| | | | | | | 3C | AE | 9F | 00110 | PUSHAB | TMP_STR | 1943 |
| | | | | | | 5B | DD | 00113 | PUSHL | R11 | | |
| | | | | | | 5B | DD | 00115 | PUSHL | R11 | | |
| | | 00000000G | | 00 | | 03 | FB | 00117 | CALLS | #3, STR\$CONCAT | | |
| | | | | | | 3C | AE | 9F | 0011E | PUSHAB | TMP_STR | 1944 |
| | | 00000000G | | 00 | | 01 | FB | 00121 | CALLS | #1, -STR\$FREE1_DX | | |
| | | | | | | FEE7 | 31 | 00128 | BRW | 1\$ | | 1945 |
| | | | 04 | AE | | 01 | B0 | 0012B | 14\$: | MOVW | #1, FORMAT_STR_LEN | 1959 |
| | | | | | | 66 | C1 | 0012F | ADDL3 | (R6), R8, R7 | | 1961 |
| 5A | FB9D | 57 | 01 | A7 | 04 | AE | 2A | 00133 | SCANC | FORMAT_STR_LEN, 1(R7), CLASS_TABLE, - | | 1960 |
| | | | | | | 02 | 12 | 0013C | BNEQ | 15\$ | | |
| | | | | | | 51 | D4 | 0013E | CLRL | R1 | | |
| | | | 54 | | | 51 | D0 | 00140 | 15\$: | MOVL | R1, CHAR_PTR | |
| | | | | | | 72 | 13 | 00143 | BEQL | 23\$ | | 1963 |
| | | | 50 | | | 69 | 3C | 00145 | MOVZWL | (R9), R0 | | 1983 |
| | | | 50 | | | 02 | C2 | 00148 | SUBL2 | #2, R0 | | |
| | | | 50 | | | 66 | D1 | 0014B | CMPL | (R6), R0 | | |
| | | | | | | 22 | 12 | 0014E | BNEQ | 18\$ | | |
| | | | | | | 66 | D5 | 00150 | TSTL | (R6) | | 1990 |
| | | | | | | 03 | 12 | 00152 | BNEQ | 16\$ | | |
| | | | | | | 01B3 | 31 | 00154 | BRW | 47\$ | | |
| | | | 66 | | | 02 | C0 | 00157 | 16\$: | ADDL2 | #2, (R6) | 1993 |
| | | | 24 | 01 | | A5 | 91 | 0015A | CMPB | 1(R5), #36 | | 1994 |
| | | | | | | 0F | 13 | 0015E | BEQL | 17\$ | | |
| | | | 2A | 01 | | A5 | 91 | 00160 | CMPB | 1(R5), #42 | | 1995 |
| | | | | | | 09 | 13 | 00164 | BEQL | 17\$ | | |
| | | | 3C | 01 | | A5 | 91 | 00166 | CMPB | 1(R5), #60 | | 1996 |
| | | | | | | 03 | 13 | 0016A | BEQL | 17\$ | | |
| | | | | | | 14 | AE | D6 | INCL | START_CNS_CNT | | 1998 |
| | | | | | | 014F | 31 | 0016F | 17\$: | BRW | 45\$ | 1985 |
| | | | | | | 40 | AA | 9E | 18\$: | MOVAB | 64(R10), UNDER_MASK | 2008 |
| 5A | FB5A | CF | 02 | A7 | 04 | AE | 2A | 00176 | SCANC | FORMAT_STR_LEN, 2(R7), CLASS_TABLE, - | | 2009 |
| | | | | | | 02 | 12 | 0017F | BNEQ | 19\$ | | |
| | | | | | | 51 | D4 | 00181 | CLRL | R1 | | |
| | | | 54 | | | 51 | D0 | 00183 | 19\$: | MOVL | R1, CHAR_PTR | |
| | | | | | | 15 | 12 | 00186 | BNEQ | 20\$ | | 2012 |
| | | | 24 | 01 | | A5 | 91 | 00188 | CMPB | 1(R5), #36 | | 2022 |
| | | | | | | 29 | 13 | 0018C | BEQL | 23\$ | | |
| | | | 2A | 01 | | A5 | 91 | 0018E | CMPB | 1(R5), #42 | | 2023 |
| | | | | | | 23 | 13 | 00192 | BEQL | 23\$ | | |
| | | | 3C | 01 | | A5 | 91 | 00194 | CMPB | 1(R5), #60 | | 2024 |
| | | | | | | 1D | 13 | 00198 | BEQL | 23\$ | | |
| | | | | | | 00BB | 31 | 0019A | BRW | 34\$ | | 2027 |
| | | | 52 | FF | | A4 | 9E | 0019D | 20\$: | MOVAB | -1(CHAR_PTR), R2 | 2044 |
| | | | 24 | | | 62 | 91 | 001A1 | CMPB | (R2), #36 | | 2047 |
| | | | | | | 06 | 12 | 001A4 | BNEQ | 21\$ | | |
| | | | 24 | 01 | | A2 | 91 | 001A6 | CMPB | 1(R2), #36 | | 2048 |

| | | | | | | | | | | | | |
|-------|----|----|------|----|-------|-------|--------|------------------------|--|--|--|------|
| | | | 09 | 11 | 001AA | | BRB | 22\$ | | | | |
| | 2A | | 62 | 91 | 001AC | 21\$: | CMPB | (R2), #42 | | | | 2057 |
| | | | 0E | 12 | 001AF | | BNEQ | 25\$ | | | | |
| | 2A | 01 | A2 | 91 | 001B1 | | CMPB | 1(R2), #42 | | | | 2058 |
| | | | 03 | 13 | 001B5 | 22\$: | BEQL | 24\$ | | | | |
| | | | 0102 | 31 | 001B7 | 23\$: | BRW | 43\$ | | | | |
| | 66 | | 03 | C0 | 001BA | 24\$: | ADDL2 | #3, (R6) | | | | 2061 |
| | | | 6A | 11 | 001BD | | BRB | 31\$ | | | | 2062 |
| 5E | 8F | | 62 | 91 | 001BF | 25\$: | CMPB | (R2), #94 | | | | 2067 |
| | | | 25 | 12 | 001C3 | | BNEQ | 28\$ | | | | |
| | 50 | | 69 | 3C | 001C5 | | MOVZWL | (R9), R0 | | | | 2068 |
| | 50 | | 05 | C2 | 001C8 | | SUBL2 | #5, R0 | | | | |
| | 50 | | 66 | D1 | 001CB | | CMPB | (R6), R0 | | | | |
| | | | 03 | 1B | 001CE | | BLEQU | 26\$ | | | | |
| | | | 00E4 | 31 | 001D0 | | BRW | 42\$ | | | | |
| 5E | 8F | 01 | A2 | 91 | 001D3 | 26\$: | CMPB | 1(R2), #94 | | | | 2071 |
| | | | 0C | 12 | 001D8 | | BNEQ | 27\$ | | | | |
| 5E | 8F | 02 | A2 | 91 | 001DA | | CMPB | 2(R2), #94 | | | | 2072 |
| | | | 7D | 12 | 001DF | | BNEQ | 36\$ | | | | |
| 5E | 8F | 03 | A2 | 91 | 001E1 | | CMPB | 3(R2), #94 | | | | 2073 |
| | | | 76 | 12 | 001E6 | 27\$: | BNEQ | 36\$ | | | | |
| | | | 69 | 11 | 001E8 | | BRB | 33\$ | | | | 2076 |
| | 3C | | 62 | 91 | 001EA | 28\$: | CMPB | (R2), #60 | | | | 2083 |
| | | | 69 | 12 | 001ED | | BNEQ | 34\$ | | | | |
| | 50 | | 69 | 3C | 001EF | | MOVZWL | (R9), R0 | | | | 2085 |
| | 50 | | 04 | C2 | 001F2 | | SUBL2 | #4, R0 | | | | |
| | 50 | | 66 | D1 | 001F5 | | CMPB | (R6), R0 | | | | |
| | | | 03 | 1B | 001F8 | | BLEQU | 29\$ | | | | |
| | | | 00B5 | 31 | 001FA | | BRW | 41\$ | | | | |
| OC | AE | | 01 | C1 | 001FD | 29\$: | ADDL3 | #1, (R6), TMP_CHAR_CNT | | | | 2092 |
| | | OC | AE | 9F | 00202 | | PUSHAB | TMP_CHAR_CNT | | | | 2093 |
| | | 1C | AE | 9F | 00205 | | PUSHAB | TMP_PARAM_BLK | | | | |
| | | | 59 | DD | 00208 | | PUSHL | R9 | | | | |
| | | | 25 | DD | 0020A | | PUSHL | #37 | | | | |
| 0000V | CF | | 04 | FB | 0020C | | CALLS | #4, SPAN_BRACKET | | | | |
| | 12 | | 50 | E8 | 00211 | | BLBS | R0, 30\$ | | | | |
| | | OC | AE | 9F | 00214 | | PUSHAB | TMP_CHAR_CNT | | | | 2096 |
| | | 1C | AE | 9F | 00217 | | PUSHAB | TMP_PARAM_BLK | | | | |
| | | | 59 | DD | 0021A | | PUSHL | R9 | | | | |
| | | | 30 | DD | 0021C | | PUSHL | #48 | | | | |
| 0000V | CF | | 04 | FB | 0021E | | CALLS | #4, SPAN_BRACKET | | | | |
| | 05 | | 50 | E9 | 00223 | | BLBC | R0, 32\$ | | | | |
| | 66 | | 04 | C0 | 00226 | 30\$: | ADDL2 | #4, (R6) | | | | 2101 |
| | | | 30 | 11 | 00229 | 31\$: | BRB | 35\$ | | | | 2102 |
| | | OC | AE | 9F | 0022B | 32\$: | PUSHAB | TMP_CHAR_CNT | | | | 2105 |
| | | 1C | AE | 9F | 0022E | | PUSHAB | TMP_PARAM_BLK | | | | |
| | | | 59 | DD | 00231 | | PUSHL | R9 | | | | |
| 0000V | 7E | 43 | 8F | 9A | 00233 | | MOVZBL | #67, -(SP) | | | | |
| | CF | | 04 | FB | 00237 | | CALLS | #4, SPAN_BRACKET | | | | |
| | 14 | | 50 | E8 | 0023C | | BLBS | R0, 33\$ | | | | |
| | | OC | AE | 9F | 0023F | | PUSHAB | TMP_CHAR_CNT | | | | 2108 |
| | | 1C | AE | 9F | 00242 | | PUSHAB | TMP_PARAM_BLK | | | | |
| | | | 59 | DD | 00245 | | PUSHL | R9 | | | | |
| 0000V | 7E | 63 | 8F | 9A | 00247 | | MOVZBL | #99, -(SP) | | | | |
| | CF | | 04 | FB | 0024B | | CALLS | #4, SPAN_BRACKET | | | | |
| | 5F | | 50 | E9 | 00250 | | BLBC | R0, 41\$ | | | | |
| | 66 | | 05 | C0 | 00253 | 33\$: | ADDL2 | #5, (R6) | | | | 2113 |

| | | | | | | | | | | |
|-----------|----|----|----------|----|-------|-------|--------|-------------------------|--|------|
| | | | 03 | 11 | 00256 | | BRB | 35\$ | | 2114 |
| | 66 | | 02 | C0 | 00258 | 34\$: | ADDL2 | #2, (R6) | | 2133 |
| | | 14 | AE | D6 | 0025B | 35\$: | INCL | START_CNS_CNT | | 2134 |
| | | | 5E | 11 | 0025E | 36\$: | BRB | 44\$ | | 1923 |
| | | | 66 | D6 | 00260 | 37\$: | INCL | (R6) | | 2153 |
| | | | 5D | 11 | 00262 | | BRB | 45\$ | | 2149 |
| | 3C | | 65 | 91 | 00264 | 38\$: | CMPB | (R5), #60 | | 2158 |
| | | | 58 | 12 | 00267 | | BNEQ | 45\$ | | |
| | 50 | | 69 | 3C | 00269 | | MOVZWL | (R9), R0 | | 2161 |
| | 50 | | 02 | C2 | 0026C | | SUBL2 | #2, R0 | | |
| | 50 | | 66 | D1 | 0026F | | CMPL | (R6), R0 | | |
| | | | 48 | 1E | 00272 | | BGEQU | 43\$ | | |
| | 50 | 01 | A5 | 9A | 00274 | | MOVZBL | 1(R5), R0 | | 2164 |
| | 25 | | 50 | 91 | 00278 | | CMPB | R0, #37 | | 2166 |
| | | | 08 | 12 | 0027B | | BNEQ | 40\$ | | |
| | 3E | 02 | A5 | 91 | 0027D | 39\$: | CMPB | 2(R5), #62 | | 2167 |
| | | | 3E | 13 | 00281 | | BEQL | 45\$ | | |
| | | | 32 | 11 | 00283 | | BRB | 42\$ | | 2171 |
| | 30 | | 50 | 91 | 00285 | 40\$: | CMPB | R0, #48 | | 2173 |
| | | | F3 | 13 | 00288 | | BEQL | 39\$ | | |
| 43 | 8F | | 50 | 91 | 0028A | | CMPB | R0, #67 | | 2180 |
| | | | 2C | 12 | 0028E | | BNEQ | 43\$ | | |
| 44 | 8F | 02 | A5 | 91 | 00290 | | CMPB | 2(R5), #68 | | 2181 |
| | | | 20 | 12 | 00295 | | BNEQ | 42\$ | | |
| | 50 | | 69 | 3C | 00297 | | MOVZWL | (R9), R0 | | 2184 |
| | 50 | | 03 | C2 | 0029A | | SUBL2 | #3, R0 | | |
| | 50 | | 66 | D1 | 0029D | | CMPL | (R6), R0 | | |
| | | | 10 | 1E | 002A0 | | BGEQU | 41\$ | | |
| | 3E | 03 | A5 | 91 | 002A2 | | CMPB | 3(R5), #62 | | 2187 |
| | | | 0A | 12 | 002A6 | | BNEQ | 41\$ | | |
| | 58 | | 55 | D1 | 002A8 | | CMPL | R5, R8 | | 2191 |
| | | | 14 | 12 | 002AB | | BNEQ | 45\$ | | |
| | 66 | | 04 | C0 | 002AD | | ADDL2 | #4, (R6) | | 2195 |
| | | | 0C | 11 | 002B0 | | BRB | 44\$ | | 2189 |
| | 66 | | 03 | C0 | 002B2 | 41\$: | ADDL2 | #3, (R6) | | 2205 |
| | | | 07 | 11 | 002B5 | | BRB | 44\$ | | 2183 |
| | 66 | | 02 | C0 | 002B7 | 42\$: | ADDL2 | #2, (R6) | | 2209 |
| | | | 02 | 11 | 002BA | | BRB | 44\$ | | 2181 |
| | | | 66 | D6 | 002BC | 43\$: | INCL | (R6) | | 2217 |
| | | | FD56 | 31 | 002BE | 44\$: | BRW | 2\$ | | 2158 |
| | 14 | AE | 66 | D1 | 002C1 | 45\$: | CMPL | (R6), START_CNS_CNT | | 2236 |
| | | | 3F | 19 | 002C5 | | BLSS | 46\$ | | |
| 56 | 66 | 14 | AE | C3 | 002C7 | | SUBL3 | START_CNS_CNT, (R6), R6 | | 2240 |
| | 10 | AE | 01 | A6 | 9E | 002CC | MOVAB | 1(R6), CNS_LENGTH | | |
| | 3C | AE | 020E0000 | 8F | D0 | 002D1 | MOVL | #34471936, -TMP_STR | | 2241 |
| | | | 40 | AE | D4 | 002D9 | CLRL | TMP_STR+4 | | 2244 |
| | | | 10 | AE | 9F | 002DC | PUSHAB | CNS_LENGTH | | 2245 |
| | | | 18 | AE | 9F | 002DF | PUSHAB | START_CNS_CNT | | |
| | | | 59 | DD | 002E2 | | PUSHL | R9 | | |
| | | 48 | AE | 9F | 002E4 | | PUSHAB | TMP_STR | | |
| 00000000G | 00 | | 04 | FB | 002E7 | | CALLS | #4, -STR\$LEN_EXTR | | |
| | | 3C | AE | 9F | 002EE | | PUSHAB | TMP_STR | | 2246 |
| | | | 5B | DD | 002F1 | | PUSHL | R11 | | |
| | | | 5B | DD | 002F3 | | PUSHL | R11 | | |
| 00000000G | 00 | | 03 | FB | 002F5 | | CALLS | #3, STR\$CONCAT | | |
| | | 3C | AE | 9F | 002FC | | PUSHAB | TMP_STR | | 2247 |
| 00000000G | 00 | | 01 | FB | 002FF | | CALLS | #1, -STR\$FREE1_DX | | |

BASS\$FOR_INT
2-013

BASS\$FORMAT INT - Basic format interpreter
SPAN_CONSTANT - find start of format sequence

D 8
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 40
(5)

```
50          01 D0 00306 46$:  MOVL  #1, R0
              04 00309          RET
              50 D4 0030A 47$:  CLRL  R0
              04 0030C          RET
```

```
: 2250
:
: 2252
:
```

: Routine Size: 781 bytes, Routine Base: _BAS\$CODE + 032D

```

: 1552 2253 1 %SBTTL 'SPAN_SINGLE_QUOTE - process single quote formatting classes'
: 1553 2254 1 ROUTINE SPAN_SINGLE_QUOTE (
: 1554 2255 1
: 1555 2256 1 P_FORMAT_STR, ! format string
: 1556 2257 1 P_PARAM_BLK, ! print using parameter block
: 1557 2258 1 P_CHAR_CNT) = ! character position within format string
: 1558 2259 1
: 1559 2260 2 BEGIN
: 1560 2261 2
: 1561 2262 2 !++
: 1562 2263 2
: 1563 2264 2 FUNCTIONAL DESCRIPTION:
: 1564 2265 2
: 1565 2266 2 This routine searches FORMAT_STR for the start of something other
: 1566 2267 2 than one of the valid single quote format classes
: 1567 2268 2 (left-justify, center-justify, right-justify, extended)
: 1568 2269 2 beginning at the CHAR_CNT + 1st character
: 1569 2270 2 position in FORMAT_STR. CHAR_CNT is then set to the last
: 1570 2271 2 character position in the formatting sequence.
: 1571 2272 2 Thus CHAR_CNT returned
: 1572 2273 2 unchanged indicates that no further formatting string was found.
: 1573 2274 2 The routine returns 0 if an illegal formatting character
: 1574 2275 2 formatting sequence is located and 1 otherwise.
: 1575 2276 2
: 1576 2277 2 This routine will be called initially to determine the end of
: 1577 2278 2 any leading constant in the format string. This routine
: 1578 2279 2 will be called after any formatting has taken place to determine
: 1579 2280 2 the end of any trailing constant in the format string.
: 1580 2281 2
: 1581 2282 2 CALLING SEQUENCE:
: 1582 2283 2
: 1583 2284 2 STATUS.wlc.v = SPAN_SINGLE_QUOTE (P_FORMAT_STR.rt.dx,
: 1584 2285 2 P_PARAM_BLK.mr.r, P_CHAR_CNT.ml.r)
: 1585 2286 2
: 1586 2287 2 FORMAL PARAMETERS:
: 1587 2288 2
: 1588 2289 2 P_FORMAT_STR - address of string descriptor for format string
: 1589 2290 2 P_PARAM_BLK - address of print using parameter block
: 1590 2291 2 P_CHAR_CNT - address of longword for character position within
: 1591 2292 2 format string; this parameter is passed as the
: 1592 2293 2 position 1 before the point at which the format
: 1593 2294 2 sequence should start; this parameter is returned
: 1594 2295 2 as the last position of the format sequence
: 1595 2296 2
: 1596 2297 2 IMPLICIT INPUTS:
: 1597 2298 2
: 1598 2299 2 NONE
: 1599 2300 2
: 1600 2301 2 IMPLICIT OUTPUTS:
: 1601 2302 2
: 1602 2303 2 NONE
: 1603 2304 2
: 1604 2305 2 COMPLETION CODES:
: 1605 2306 2
: 1606 2307 2 1 - routine completed successfully
: 1607 2308 2
: 1608 2309 2 SIDE EFFECTS:

```

1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665

2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366

```

NONE
EXAMPLES:
      FORMAT_STR      CHAR_CNT (in)  CHAR_CNT (out)
-----
abc'cccdef'cc      4              7
abc'llldef'cc      4              7
abc'rrrdef'cc      4              7
abc'cccdef*        4              7
ab'lab             3              4
ab'                3              3
ab'z               3              3
--
BIND
PARAM_BLK = (.P PARAM_BLK) : BLOCK [, BYTE],
FORMAT_STR = (.P FORMAT_STR) : BLOCK [, BYTE],
CHAR_CNT = (.P CHAR_CNT),
CHAR = (.FORMAT_STR[DSC$A_POINTER] + .CHAR_CNT) : BYTE;

LOCAL
CHAR_PTR;                ! pointer to possible start point
                          ! for formatting sequence

IF .CHAR_CNT EQLU .FORMAT_STR [DSC$W_LENGTH]
THEN                        ! This is the case of a format string
                          ! with just a single quote
    BEGIN
    PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK;
    PARAM_BLK [CHARACTER] = 1;
    RETURN 1
    END;

SELECTONEU .CHAR OF
SET
    [CHAR_L, CHAR_LOWER_L] :
    BEGIN
    !+
    !- Span L and l.
    SPAN_TEXT (MASK L, FORMAT_STR, CHAR_CNT,
               PARAM_BLK [CHARACTER]);
    PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK
    END;
    [CHAR_C, CHAR_LOWER_C] :
    BEGIN
    !+
    !- Span C and c.

```

```

: 1666 2367
: 1667 2368     SPAN_TEXT (MASK_C, FORMAT_STR, CHAR_CNT,
: 1668 2369         PARAM_BLK [CHARACTER]);
: 1669 2370     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR C_JUSTIFY_MASK
: 1670 2371     END;
: 1671 2372
: 1672 2373     [CHAR_R, CHAR_LOWER_R] :
: 1673 2374     BEGIN
: 1674 2375
: 1675 2376     !+
: 1676 2377     !- Span R and r.
: 1677 2378
: 1678 2379
: 1679 2380     SPAN_TEXT (MASK_R, FORMAT_STR, CHAR_CNT,
: 1680 2381         PARAM_BLK [CHARACTER]);
: 1681 2382     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR R_JUSTIFY_MASK
: 1682 2383     END;
: 1683 2384
: 1684 2385     [CHAR_E, CHAR_LOWER_E] :
: 1685 2386     BEGIN
: 1686 2387
: 1687 2388     !+
: 1688 2389     !- Span E and e.
: 1689 2390
: 1690 2391
: 1691 2392     SPAN_TEXT (MASK_E, FORMAT_STR, CHAR_CNT,
: 1692 2393         PARAM_BLK [CHARACTER]);
: 1693 2394     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR EXTEND_MASK
: 1694 2395     END;
: 1695 2396
: 1696 2397     [CHAR_UNDERSCORE]:
: 1697 2398     !+
: 1698 2399     !- Underscore causes the next format character to be ignored. If not
: 1699 2400     !- followed by a valid format character, it has no effect. Underscore is
: 1700 2401     !- handled in SPAN_CONSTANT so that the format character may be appended
: 1701 2402     !- as a constant in the output string.
: 1702 2403
: 1703 2404     BEGIN
: 1704 2405     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK;
: 1705 2406     PARAM_BLK [CHARACTER] = 1;
: 1706 2407     RETURN 1
: 1707 2408     END;
: 1708 2409
: 1709 2410     [OTHERWISE] :
: 1710 2411     BEGIN
: 1711 2412
: 1712 2413     !+
: 1713 2414     !- Just a single quote followed by a constant string.
: 1714 2415
: 1715 2416
: 1716 2417     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK;
: 1717 2418     PARAM_BLK [CHARACTER] = 1
: 1718 2419     END;
: 1719 2420
: 1720 2421     TES;
: 1721 2422
: 1722 2423     RETURN 1                                ! return on success

```

: 1723
: 1724

2424 2
2425 1 END;

: End of routine SPAN_SINGLE_QUOTE

007C 00000 SPAN_SINGLE QUOTE:

| | | | | | | | | | |
|----|----|-------|----|----|-------|--------|---------------------|-------|------|
| | | | | | | .WORD | Save R2,R3,R4,R5,R6 | | 2254 |
| | 56 | 0000V | CF | 9E | 00002 | MOVAB | SPAN TEXT, R6 | | 2328 |
| | 52 | 08 | AC | DO | 00007 | MOVL | P_PARAM_BLK, R2 | | 2329 |
| | 55 | 04 | AC | DO | 0000B | MOVL | P_FORMAT_STR, R5 | | 2330 |
| | 54 | 0C | AC | DO | 0000F | MOVL | P_CHAR_CNT, R4 | | 2331 |
| 64 | | 53 | 04 | A5 | 64 | ADDL3 | (R4), 4(R5), R3 | | 2337 |
| | 10 | | | 00 | ED | CMPZV | #0, #16, (R5), (R4) | | 2349 |
| | 4C | 8F | | 78 | 13 | BEQL | 8\$ | | 2357 |
| | 6C | 8F | | 63 | 91 | CMPB | (R3), #76 | | 2358 |
| | | | | 06 | 13 | BEQL | 1\$ | | 2361 |
| | | | | 63 | 91 | CMPB | (R3), #108 | | 2369 |
| | | | | 12 | 12 | BNEQ | 2\$ | | 2370 |
| | | | 18 | A2 | 9F | PUSHAB | 24(R2) | | 2373 |
| | | | | 54 | DD | PUSHL | R4 | | 2381 |
| | | | | 55 | DD | PUSHL | R5 | | 2382 |
| | | | | 02 | DD | PUSHL | #2 | | 2385 |
| | 05 | 66 | | 04 | FB | CALLS | #4, SPAN TEXT | | 2393 |
| | A2 | | | 08 | 88 | BISB2 | #8, 5(R2) | | |
| | 43 | 8F | | 62 | 11 | BRB | 9\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #67 | | |
| | 63 | 8F | | 06 | 13 | BEQL | 3\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #99 | | |
| | | | | 12 | 12 | BNEQ | 4\$ | | |
| | | | 18 | A2 | 9F | PUSHAB | 24(R2) | | |
| | | | | 54 | DD | PUSHL | R4 | | |
| | | | | 55 | DD | PUSHL | R5 | | |
| | | | | 04 | DD | PUSHL | #4 | | |
| | 05 | 66 | | 04 | FB | CALLS | #4, SPAN TEXT | | |
| | A2 | | | 02 | 88 | BISB2 | #2, 5(R2) | | |
| | 52 | 8F | | 44 | 11 | BRB | 9\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #82 | | |
| | 72 | 8F | | 06 | 13 | BEQL | 5\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #114 | | |
| | | | | 12 | 12 | BNEQ | 6\$ | | |
| | | | 18 | A2 | 9F | PUSHAB | 24(R2) | | |
| | | | | 54 | DD | PUSHL | R4 | | |
| | | | | 55 | DD | PUSHL | R5 | | |
| | | | | 08 | DD | PUSHL | #8 | | |
| | 05 | 66 | | 04 | FB | CALLS | #4, SPAN TEXT | | |
| | A2 | | | 01 | 88 | BISB2 | #1, 5(R2) | | |
| | 45 | 8F | | 26 | 11 | BRB | 9\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #69 | | |
| | 65 | 8F | | 06 | 13 | BEQL | 7\$ | | |
| | | | | 63 | 91 | CMPB | (R3), #101 | | |
| | | | | 12 | 12 | BNEQ | 8\$ | | |
| | | | 18 | A2 | 9F | PUSHAB | 24(R2) | | |
| | | | | 54 | DD | PUSHL | R4 | | |
| | | | | 55 | DD | PUSHL | R5 | | |
| | | | | 10 | DD | PUSHL | #16 | | |

BAS\$\$FOR_INT
2-013

BAS\$\$FORMAT_INT - Basic format interpreter
SPAN_SINGLE_QUOTE - process single quote format

I 8
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 45
(6)

| | | | | | |
|----|----|----|---------------|-------|---------------|
| 05 | 66 | 04 | FB 0008E | CALLS | #4, SPAN TEXT |
| | A2 | 04 | 88 00091 | BISB2 | #4, 5(R2) |
| | | 08 | 11 00095 | BRB | 9\$ |
| 05 | A2 | 08 | 88 00097 8\$: | BISB2 | #8, 5(R2) |
| 18 | A2 | 01 | D0 0009B | MOVL | #1, 24(R2) |
| | 50 | 01 | D0 0009F 9\$: | MOVL | #1, R0 |
| | | 04 | 000A2 | RET | |

:
: 2394
:
: 2417
:
: 2418
:
: 2423
:
: 2425

; Routine Size: 163 bytes, Routine Base: _BAS\$CODE + 063A

```

: 1726      2426  1 %SBTTL 'SPAN_TEXT - span various text formatting classes'
: 1727      2427  1 ROUTINE SPAN_TEXT (
: 1728      2428  1
: 1729      2429  1     P_SPAN_MASK,           | mask for SPANC instruction
: 1730      2430  1     P_FORMAT_STR,         | format string
: 1731      2431  1     P_CHAR_CNT,           | position withing format string
: 1732      2432  1     P_FORMAT_LENGTH) = | length of format sequence
: 1733      2433  1
: 1734      2434  2 BEGIN
: 1735      2435  2
: 1736      2436  2 !++
: 1737      2437  2
: 1738      2438  2 FUNCTIONAL DESCRIPTION:
: 1739      2439  2
: 1740      2440  2     This routine spans a particular class of text formatting characters
: 1741      2441  2     based on SPAN_MASK.
: 1742      2442  2
: 1743      2443  2 CALLING SEQUENCE:
: 1744      2444  2
: 1745      2445  2     STATUS.wlc.v = SPAN_TEXT (P_SPAN_MASK, P_FORMAT_STR.rt.dx,
: 1746      2446  2     P_CHAR_CNT.ml.r, P_FORMAT_LENGTH.wl.r)
: 1747      2447  2
: 1748      2448  2 FORMAL PARAMETERS:
: 1749      2449  2
: 1750      2450  2     P_SPAN_MASK           - address of a byte with the mask for the spanc
: 1751      2451  2     instruction
: 1752      2452  2     P_FORMAT_STR         - address of string descriptor for format string
: 1753      2453  2     P_CHAR_CNT           - address of longword for character position within
: 1754      2454  2     format string; this paramater is passed as the
: 1755      2455  2     position 2 before the point at which the format
: 1756      2456  2     sequence scanc should start;
: 1757      2457  2     this parameter is returned
: 1758      2458  2     as the last position of the format sequence
: 1759      2459  2     P_FORMAT_LENGTH - address of longword for format length
: 1760      2460  2
: 1761      2461  2 IMPLICIT INPUTS:
: 1762      2462  2
: 1763      2463  2     NONE
: 1764      2464  2
: 1765      2465  2 IMPLICIT OUTPUTS:
: 1766      2466  2
: 1767      2467  2     NONE
: 1768      2468  2
: 1769      2469  2 COMPLETION CODES:
: 1770      2470  2
: 1771      2471  2     1                     - routine completed successfully
: 1772      2472  2
: 1773      2473  2 SIDE EFFECTS:
: 1774      2474  2
: 1775      2475  2     NONE
: 1776      2476  2
: 1777      2477  2 EXAMPLES:
: 1778      2478  2
: 1779      2479  2     FORMAT_STR          CHAR_CNT (in)  CHAR_CNT (out)  FORMAT_LENGTH
: 1780      2480  2     -----            - - - - -      - - - - -      - - - - -
: 1781      2481  2     abc'cccdef'cc      4              7              4
: 1782      2482  2     abc'llldef'cc      4              7              4

```


| | | | | | | | | | | |
|----|----|----|----|-------|-------|-------|-------|-------------------------|-----------------------------------|------|
| | 50 | | 51 | D0 | 00024 | 1\$: | MOVL | R1, CHAR_PTR | | |
| | | | 0A | 13 | 00027 | | BEQL | 2\$ | | 2506 |
| | 50 | | 54 | C2 | 00029 | | SUBL2 | R4, R0 | | 2508 |
| 10 | BC | 01 | A0 | 9E | 0002C | | MOVAB | 1(R0), @P_FORMAT_LENGTH | | |
| | | | 13 | 11 | 00031 | | BRB | 4\$ | | |
| | 50 | 01 | A5 | 9E | 00033 | 2\$: | MOVAB | 1(R5), R0 | | 2511 |
| | 02 | | 50 | D1 | 00037 | | CMPL | R0, #2 | | |
| | | | 06 | 1B | 0003A | | BLEQU | 3\$ | | |
| 10 | BC | | 50 | D0 | 0003C | | MOVL | R0, @P_FORMAT_LENGTH | | 2513 |
| | | | 04 | 11 | 00040 | | BRB | 4\$ | | |
| 10 | BC | | 02 | D0 | 00042 | 3\$: | MOVL | #2, @P_FORMAT_LENGTH | | 2517 |
| 50 | 0C | BC | 10 | BC | C1 | 00046 | 4\$: | ADDL3 | @P_FORMAT_LENGTH, @P_CHAR_CNT, R0 | 2519 |
| | 0C | BC | FF | A0 | 9E | 0004C | | MOVAB | -1(R0), @P_CHAR_CNT | |
| | | | 01 | D0 | 00051 | | MOVL | #1, R0 | | 2521 |
| | | | 04 | 00054 | | | RET | | | 2523 |

; Routine Size: 85 bytes, Routine Base: _BAS\$CODE + 06DD

```

: 1825      2524 1 %SBTTL 'SPAN_BACKSLASH - process backslash formatting class'
: 1826      2525 1 ROUTINE SPAN_BACKSLASH (
: 1827      2526 1
: 1828      2527 1     P_FORMAT_STR,  ! format string
: 1829      2528 1     P_PARAM_BLK,  ! print using parameter block
: 1830      2529 1     P_CHAR_CNT) = ! character position within format string
: 1831      2530 1
: 1832      2531 2 BEGIN
: 1833      2532 2
: 1834      2533 2  !++
: 1835      2534 2
: 1836      2535 2     FUNCTIONAL DESCRIPTION:
: 1837      2536 2
: 1838      2537 2     This routine finds the end of a backslash formatting sequence
: 1839      2538 2     which consists of a backslash followed by some number of blanks (>= 0)
: 1840      2539 2     followed by a backslash. The initial backslash has been located
: 1841      2540 2     when this routine is called. An error is returned if something
: 1842      2541 2     other than a proper backslash formatting sequence is located.
: 1843      2542 2
: 1844      2543 2     CALLING SEQUENCE:
: 1845      2544 2
: 1846      2545 2     STATUS.wlc.v = SPAN_BACKSLASH (P_FORMAT_STR.rt.dx,
: 1847      2546 2     P_PARAM_BLK.mr.r, P_CHAR_CNT.ml.r)
: 1848      2547 2
: 1849      2548 2     FORMAL PARAMETERS:
: 1850      2549 2
: 1851      2550 2     P_FORMAT_STR - address of string descriptor for format string
: 1852      2551 2     P_PARAM_BLK - address of print using parameter block
: 1853      2552 2     P_CHAR_CNT - address of longword for character position within
: 1854      2553 2     format string; this parameter is passed as the
: 1855      2554 2     position 1 before the point at which the format
: 1856      2555 2     sequence should start; this parameter is returned
: 1857      2556 2     as the last position of the format sequence
: 1858      2557 2
: 1859      2558 2     IMPLICIT INPUTS:
: 1860      2559 2
: 1861      2560 2     NONE
: 1862      2561 2
: 1863      2562 2     IMPLICIT OUTPUTS:
: 1864      2563 2
: 1865      2564 2     NONE
: 1866      2565 2
: 1867      2566 2     COMPLETION CODES:
: 1868      2567 2
: 1869      2568 2     1 - routine completed successfully (format sequence
: 1870      2569 2     for backslash located)
: 1871      2570 2     0 - no backslash format sequence located
: 1872      2571 2
: 1873      2572 2     SIDE EFFECTS:
: 1874      2573 2
: 1875      2574 2     NONE
: 1876      2575 2
: 1877      2576 2     EXAMPLES:
: 1878      2577 2
: 1879      2578 2     FORMAT_STR CHAR_CNT (in) CHAR_CNT (out) return value
: 1880      2579 2     -----
: 1881      2580 2     abc\def 4 5 1

```

```

: 1882      2581 2 |      abc\  \def  4      8      1
: 1883      2582 2 |      abc\def  4      4      0
: 1884      2583 2 |      abc\  def  4      4      0
: 1885      2584 2 |      abc\  <eos>  4      4      0
: 1886      2585 2 |
: 1887      2586 2 |  --
: 1888      2587 2 |
: 1889      2588 2 | BIND
: 1890      2589 2 |      PARAM_BLK = (.P PARAM_BLK) : BLOCK [, BYTE],
: 1891      2590 2 |      FORMAT_STR = (.P FORMAT_STR) : BLOCK [, BYTE],
: 1892      2591 2 |      CHAR_CNT = (.P_CHAR_CNT);
: 1893      2592 2 |
: 1894      2593 2 | LOCAL
: 1895      2594 2 |      CHAR_PTR;
: 1896      2595 2 |
: 1897      2596 2 |
: 1898      2597 2 | PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR L_JUSTIFY_MASK;
: 1899      2598 2 |
: 1900      2599 2 | !+
: 1901      2600 2 | ! Span blanks; next character should then be another backslash.
: 1902      2601 2 | !-
: 1903      2602 2 |
: 1904      2603 2 | CHAR_PTR = CH$FIND NOT CH (.FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT,
: 1905      2604 2 |      .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT, CHAR_SPACE);
: 1906      2605 2 |
: 1907      2606 2 | !+
: 1908      2607 2 | ! CHAR_PTR should point to one past the last blank if a blank terminator
: 1909      2608 2 | ! was found; otherwise CHAR_PTR is the null pointer.
: 1910      2609 2 |
: 1911      2610 2 | ! In VAX-11 Bliss-32 V3, SKPC should be a BUILTIN allowing output registers
: 1912      2611 2 | ! which could make the computation of the number of characters in the
: 1913      2612 2 | ! formatting sequence more efficient.
: 1914      2613 2 | !-
: 1915      2614 2 |
: 1916      2615 2 | IF CH$FAIL (.CHAR_PTR)
: 1917      2616 2 | THEN
: 1918      2617 2 |      RETURN 0
: 1919      2618 2 |
: 1920      2619 2 |
: 1921      2620 2 |
: 1922      2621 2 | ELSE
: 1923      2622 2 |      BEGIN
: 1924      2623 2 |      BIND CHARS = (.CHAR_PTR) : VECTOR [, BYTE];
: 1925      2624 2 |
: 1926      2625 2 | !+
: 1927      2626 2 | ! Make sure that the first character past the blanks is a backslash.
: 1928      2627 2 | !-
: 1929      2628 2 |
: 1930      2629 2 |      IF .CHARS [0] EQLU CHAR_BACKSLASH
: 1931      2630 2 |      THEN
: 1932      2631 2 |          BEGIN
: 1933      2632 2 |          PARAM_BLK [CHARACTER] = .CHAR_PTR -
: 1934      2633 2 |          (.FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT) + 2;
: 1935      2634 2 |          CHAR_CNT = .CHAR_CNT + .PARAM_BLK [CHARACTER] - 1;
: 1936      2635 2 |          RETURN 1;
: 1937      2636 2 |          ! this return gets the case abc\  \def
: 1938      2637 2 |          ! or the case abc\\def
:          END

```

```

: 1939      2638 4
: 1940      2639      ELSE
: 1941      2640      RETURN 0;      ! this return gets the case abc\ def
: 1942      2641
: 1943      2642      END
: 1944      2643
: 1945      2644 1 END;      ! End of routine SPAN_BACKSLASH

```

```

                                000C 00000 SPAN_BACKSLASH:
                                .WORD      Save R2,R3
                                52      08      AC      D0      00002      MOVL      P_PARAM_BLK, R2      : 2525
                                50      04      AC      D0      00006      MOVL      P_FORMAT_STR, R0      : 2589
                                05      A2      08      88      0000A      BISB2     #8, 5(R2)      : 2590
                                51      60      3C      0000E      MOVZWL   (R0), R1      : 2597
                                53      04      0C      BC      C2      00011      SUBL2    @P_CHAR_CNT, R1      : 2603
                                63      A0      0C      BC      C1      00015      ADDL3    @P_CHAR_CNT, 4(R0), R3
                                51      20      3B      0001B      SKPC     #32, R1, (R3)
                                02      12      0001F      BNEQ     1$
                                51      D4      00021      CLRL     R1
                                51      D5      00023      TSTL     CHAR_PTR      : 2604
                                1D      13      00025      BEQL     2$      : 2603
                                5C      8F      61      91      00027      CMPB     (CHAR_PTR), #92
                                17      12      0002B      BNEQ     2$      : 2615
                                51      53      C2      0002D      SUBL2    R3, R1      : 2629
                                18      A2      02      A1      9E      00030      MOVAB    2(R1), 24(R2)
                                50      0C      BC      18      A2      C1      00035      ADDL3    24(R2), @P_CHAR_CNT, R0
                                0C      BC      FF      A0      9E      0003B      MOVAB    -1(R0), @P_CHAR_CNT
                                50      01      D0      00040      MOVL     #1, R0      : 2633
                                04      00043      RET
                                50      D4      00044      CLRL     R0      : 2634
                                04      00046      RET      : 2640
                                : 2644

```

: Routine Size: 71 bytes, Routine Base: _BAS\$CODE + 0732

```

: 1947 2645 1 %SBTTL 'SPAN_BRACKET - process < > formatting classes'
: 1948 2646 1 ROUTINE SPAN_BRACKET (
: 1949 2647 1
: 1950 2648 1     FORMAT CHAR,      ! format character within < >
: 1951 2649 1     P_FORMAT_STR,    ! format string
: 1952 2650 1     P_PARAM_BLK,    ! print using parameter block
: 1953 2651 1     P_CHAR_CNT) = ! character position within format string
: 1954 2652 1
: 1955 2653 1 BEGIN
: 1956 2654 1
: 1957 2655 1 !++
: 1958 2656 1
: 1959 2657 1     FUNCTIONAL DESCRIPTION:
: 1960 2658 1
: 1961 2659 1     This routine looks for a valid '< >' sequence. Since some diagraphs
: 1962 2660 1     precede format characters and some follow, the calling program must
: 1963 2661 1     specify what particular sequence is valid at that point, and only that
: 1964 2662 1     sequence is searched for.
: 1965 2663 1
: 1966 2664 1     If a '$$' or '**' has been found, then if '<%>' or '<0>' is
: 1967 2665 1     present, it must precede all other characters. So this routine should be
: 1968 2666 1     called before SPAN_NUMERIC finishes processing the format string.
: 1969 2667 1
: 1970 2668 1     SPAN_NUMERIC will also call this routine to check for '<CD>' at the end
: 1971 2669 1     of a format string.
: 1972 2670 1
: 1973 2671 1     CALLING SEQUENCE:
: 1974 2672 1
: 1975 2673 1     STATUS.wlc.v = SPAN_BRACKET (FORMAT_CHAR.rt.r, P_FORMAT_STR.rt.dx,
: 1976 2674 1     P_PARAM_BLK.mr.r, P_CHAR_CNT.ml.r)
: 1977 2675 1
: 1978 2676 1     FORMAL PARAMETERS:
: 1979 2677 1
: 1980 2678 1     FORMAT_CHAR - address of format character within < >
: 1981 2679 1     P_FORMAT_STR - address of string descriptor for format string
: 1982 2680 1     P_PARAM_BLK - address of print using parameter block
: 1983 2681 1     P_CHAR_CNT - address of longword for character position within
: 1984 2682 1     format string; this parameter is passed as the
: 1985 2683 1     position 1 before the point at which the format
: 1986 2684 1     sequence should start; this parameter is returned
: 1987 2685 1     as the last position of the format sequence
: 1988 2686 1
: 1989 2687 1     IMPLICIT INPUTS:
: 1990 2688 1
: 1991 2689 1     NONE
: 1992 2690 1
: 1993 2691 1     IMPLICIT OUTPUTS:
: 1994 2692 1
: 1995 2693 1     NONE
: 1996 2694 1
: 1997 2695 1     COMPLETION CODES:
: 1998 2696 1
: 1999 2697 1     1 - routine completed successfully (format sequence
: 2000 2698 1     located)
: 2001 2699 1     0 - no format sequence located
: 2002 2700 1
: 2003 2701 1     SIDE EFFECTS:

```

```

2004 2702 2 |
2005 2703 2 |      NONE
2006 2704 2 |
2007 2705 2 |  EXAMPLES:
2008 2706 2 |
2009 2707 2 |      FORMAT_STR      CHAR_CNT (in)  CHAR_CNT (out)  return value
2010 2708 2 |      -----      -
2011 2709 2 |
2012 2710 2 |      $$<%>###.##      5          5          1
2013 2711 2 |      **##.##          5          5          0
2014 2712 2 |      $$<0>###,###.##-  5          5          1
2015 2713 2 |      $$###,###.##-    5          5          0
2016 2714 2 |  --
2017 2715 2 |
2018 2716 2 |  BIND
2019 2717 2 |      PARAM_BLK = (.P PARAM_BLK) : BLOCK [, BYTE],
2020 2718 2 |      FORMAT_STR = (.P FORMAT_STR) : BLOCK [, BYTE],
2021 2719 2 |      CHAR_CNT = (.P CHAR_CNT);
2022 2720 2 |
2023 2721 2 |  LOCAL
2024 2722 2 |      SUPPRESS_STR,
2025 2723 2 |      LEADING_STR,
2026 2724 2 |      FOUND_PTR,
2027 2725 2 |      CD_STR;
2028 2726 2 |
2029 2727 2 |  !+
2030 2728 2 |  ! Future formatting characters will be of the form '<x>'. This
2031 2729 2 |  ! routine is set up to allow new characters to be added easily.
2032 2730 2 |  !-
2033 2731 2 |
2034 2732 2 |  SELECTONEU .FORMAT_CHAR OF
2035 2733 2 |  SET
2036 2734 2 |  [CHAR_PERCENT]:
2037 2735 2 |  !+
2038 2736 2 |  ! Suppress a zero value.
2039 2737 2 |  !-
2040 2738 2 |      BEGIN
2041 2739 2 |      SUPPRESS_STR = %ASCIZ'<%>';
2042 2740 2 |      FOUND_PTR = CH$FIND_SUB (3,
2043 2741 2 |                          3, FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT,
2044 2742 2 |                          3, CH$PTR (SUPPRESS_STR));
2045 2743 2 |      IF (NOT CH$FAIL (.FOUND_PTR))
2046 2744 2 |      THEN
2047 2745 2 |          BEGIN
2048 2746 2 |          PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR PERCENT_MASK;
2049 2747 2 |          CHAR_CNT = .CHAR_CNT + 3;
2050 2748 2 |          PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS] + 1;
2051 2749 2 |          RETURN 1;          ! '<%>' was found
2052 2750 2 |          END;
2053 2751 2 |      END;
2054 2752 2 |
2055 2753 2 |  [CHAR_ZERO]:
2056 2754 2 |  !+
2057 2755 2 |  ! Print leading zeroes.
2058 2756 2 |  !-
2059 2757 2 |      BEGIN
2060 2758 2 |      LEADING_STR = %ASCIZ'<0>';

```

```

2061 2759 FOUND_PTR = CH$FIND_SUB (3,
2062 2760     .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT,
2063 2761     3, CH$PTR (LEADING_STR));
2064 2762 IF (NOT CH$FAIL (.FOUND_PTR))
2065 2763 THEN
2066 2764     BEGIN
2067 2765     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR ZERO_MASK;
2068 2766     CHAR_CNT = .CHAR_CNT + 3;
2069 2767     PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS] + 1;
2070 2768     RETURN 1;
2071 2769     ! '<0>' was found
2072 2770     END;
2073 2771
2074 2772 [CHAR_C, CHAR_LOWER_C]:
2075 2773 BEGIN
2076 2774     !+
2077 2775     !- Credit/debit.
2078 2776
2079 2777     !+
2080 2778     !- First, check for <CD>.
2081 2779
2082 2780     CD_STR = %ASCII'<CD>';
2083 2781     FOUND_PTR = CH$FIND_SUB (4,
2084 2782     .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT,
2085 2783     4, CH$PTR (CD_STR));
2086 2784 IF (NOT CH$FAIL (.FOUND_PTR))
2087 2785 THEN
2088 2786     BEGIN
2089 2787     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR CD_MASK;
2090 2788     CHAR_CNT = .CHAR_CNT + 4;
2091 2789     PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS] + 1;
2092 2790     RETURN 1;
2093 2791     END;
2094 2792
2095 2793     !+
2096 2794     !- Then check for <cd>.
2097 2795
2098 2796     CD_STR = %ASCII'<cd>';
2099 2797     FOUND_PTR = CH$FIND_SUB (4,
2100 2798     .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT,
2101 2799     4, CH$PTR (CD_STR));
2102 2800 IF (NOT CH$FAIL (.FOUND_PTR))
2103 2801 THEN
2104 2802     BEGIN
2105 2803     PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR CD_MASK;
2106 2804     CHAR_CNT = .CHAR_CNT + 4;
2107 2805     PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS] + 1;
2108 2806     RETURN 1;
2109 2807     END;
2110 2808
2111 2809     END;
2112 2810
2113 2811 [OTHERWISE]:
2114 2812     !+
2115 2813     !- Should never reach here.
2116 2814
2117 2815     RETURN 0;

```


BAS\$\$FOR_INT
2-013

BAS\$\$FORMAT_INT - Basic format interpreter
SPAN_BRACKET - process < > formatting classes

G 9
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 56
(9)

| | | | | | | | | | | |
|----|----|----|----|-------|-------|-------|-------|------------|--|------|
| | | | 03 | 13 | 000A4 | | BEQL | 8\$ | | |
| | 53 | | 04 | D0 | 000A6 | | MOVL | #4, R3 | | : |
| | 53 | | 04 | C2 | 000A9 | 8\$: | SUBL2 | #4, R3 | | : |
| | | | 0F | 13 | 000AC | 9\$: | BEQL | 12\$ | | : |
| 05 | A4 | 40 | 8F | 88 | 000AE | 10\$: | BISB2 | #64, 5(R4) | | 2800 |
| | 65 | | 04 | C0 | 000B3 | | ADDL2 | #4, (R5) | | 2803 |
| | | 14 | A4 | D6 | 000B6 | 11\$: | INCL | 20(R4) | | 2804 |
| | 50 | | 01 | D0 | 000B9 | | MOVL | #1, R0 | | 2805 |
| | | | | 04 | 000BC | | RET | | | 2806 |
| | | | 50 | D4 | 000BD | 12\$: | CLRL | R0 | | : |
| | | | 04 | 000BF | | | RET | | | 2824 |
| | | | | | | | | | | : |

; Routine Size: 192 bytes, Routine Base: _BAS\$CODE + 0779

```

: 2128      2825  1  XSBTTL 'SPAN_NUMERIC - processing numeric format class'
: 2129      2826  1  ROUTINE SPAN_NUMERIC (
: 2130      2827  1
: 2131      2828  1      P_SPAN_MODE,      | numeric formatting mode
: 2132      2829  1      P_FORMAT_STR,      | format string
: 2133      2830  1      P_CHAR_CNT,      | position within format string
: 2134      2831  1      P_PARAM_BLK) = | print using parameter block
: 2135      2832  1
: 2136      2833  2  BEGIN
: 2137      2834  2
: 2138      2835  2  !++
: 2139      2836  2
: 2140      2837  2  FUNCTIONAL DESCRIPTION:
: 2141      2838  2
: 2142      2839  2      This routine handles various portions of a numeric formatting
: 2143      2840  2      sequence based on SPAN_MODE. CHAR_CNT will be returned as the last
: 2144      2841  2      character of the particular portion of the numeric formatting
: 2145      2842  2      sequence under consideration. This routine will call itself
: 2146      2843  2      recursively. NUMERIC_MASK should be set in the main driving
: 2147      2844  2      routine before a call to this routine.
: 2148      2845  2
: 2149      2846  2  CALLING SEQUENCE:
: 2150      2847  2
: 2151      2848  2      STATUS.wlc.v = SPAN_NUMERIC (P_SPAN_MODE.rl.r, P_FORMAT_STR.rt.dx,
: 2152      2849  2      P_CHAR_CNT.ml.r, P_PARAM_BLK.mr.r)
: 2153      2850  2
: 2154      2851  2  FORMAL PARAMETERS:
: 2155      2852  2
: 2156      2853  2      P_SPAN_MODE      - address of longword for type of numeric span
: 2157      2854  2      P_FORMAT_STR    - address of string descriptor for format string
: 2158      2855  2      P_CHAR_CNT      - address of longword for character position within
: 2159      2856  2      format string; this parameter is passed as the
: 2160      2857  2      position 1 before the point at which the format
: 2161      2858  2      sequence should start; this parameter is passed
: 2162      2859  2      as the position 1 before the starting character
: 2163      2860  2      position of the format sequence
: 2164      2861  2      P_PARAM_BLK    - address of print using parameter block
: 2165      2862  2
: 2166      2863  2  IMPLICIT INPUTS:
: 2167      2864  2
: 2168      2865  2      DIGIT_SEP_DESC - descriptor for digit group separator
: 2169      2866  2
: 2170      2867  2  IMPLICIT OUTPUTS:
: 2171      2868  2
: 2172      2869  2      NONE
: 2173      2870  2
: 2174      2871  2  COMPLETION CODES:
: 2175      2872  2
: 2176      2873  2      1              - routine completed successfully
: 2177      2874  2
: 2178      2875  2  SIDE EFFECTS:
: 2179      2876  2
: 2180      2877  2      NONE
: 2181      2878  2
: 2182      2879  2  !--
: 2183      2880  2
: 2184      2881  2  BIND

```

```

: 2185      2882      2      SPAN MODE = (P_SPAN MODE),
: 2186      2883      2      FORMAT STR = (.P FORMAT STR) : BLOCK [, BYTE],
: 2187      2884      2      PARAM_BLK = (.P PARAM B[K]) : BLOCK [, BYTE],
: 2188      2885      2      CHAR_CNT = (.P_CHAR_CNT);
: 2189      2886      2
: 2190      2887      2      MAP
: 2191      2888      2      DIGIT_SEP_DESC : BLOCK [, BYTE];
: 2192      2889      2
: 2193      2890      2      LOCAL
: 2194      2891      2      E_FLAG,          ! 1 if E formatting is allowed; 0 otherwise
: 2195      2892      2      PERIOD_MODE,       ! fraction mode (either E or NO_E)
: 2196      2893      2      INTEGER_MODE,      ! 1 if integer mode; 0 otherwise (fraction mode)
: 2197      2894      2      TMP_LEN,          ! temporary length for format sequence
: 2198      2895      2      CHAR_PTR;         ! pointer to character in format string
: 2199      2896      2
: 2200      2897      2      CASE .SPAN_MODE FROM K_SPAN_INTEGER TO K_SPAN_FRACTION_NO_E OF
: 2201      2898      2      SET
: 2202      2899      2
: 2203      2900      2      [K_SPAN_INTEGER] :
: 2204      2901      2      BEGIN
: 2205      2902      2      E_FLAG = 1;
: 2206      2903      2      PERIOD_MODE = K_SPAN_FRACTION;
: 2207      2904      2      INTEGER_MODE = T
: 2208      2905      2      END;
: 2209      2906      2
: 2210      2907      2      [K_SPAN_INTEGER_NO_E] :
: 2211      2908      2      BEGIN
: 2212      2909      2      E_FLAG = 0;
: 2213      2910      2      PERIOD_MODE = K_SPAN_FRACTION_NO_E;
: 2214      2911      2      INTEGER_MODE = T
: 2215      2912      2      END;
: 2216      2913      2
: 2217      2914      2      [K_SPAN_FRACTION] :
: 2218      2915      2      BEGIN
: 2219      2916      2      E_FLAG = 1;
: 2220      2917      2      INTEGER_MODE = 0
: 2221      2918      2      END;
: 2222      2919      2
: 2223      2920      2      [K_SPAN_FRACTION_NO_E] :
: 2224      2921      2      BEGIN
: 2225      2922      2      E_FLAG = 0;
: 2226      2923      2      INTEGER_MODE = 0
: 2227      2924      2      END;
: 2228      2925      2
: 2229      2926      2      [OUTRANGE] :
: 2230      2927      2
: 2231      2928      2      !+
: 2232      2929      2      !- This should never happen.
: 2233      2930      2
: 2234      2931      2
: 2235      2932      2      RETURN 0          ! return here is an internal error
: 2236      2933      2
: 2237      2934      2      TES;
: 2238      2935      2
: 2239      2936      2      !+
: 2240      2937      2      !- Span pound signs.
: 2241      2938      2

```

```

: 2242 2939 2 ! In VAX-11 Bliss-32 V3, SKPC should be a BUILTIN allowing output registers
: 2243 2940 2 ! which could make the computation of the number of charaters in the
: 2244 2941 2 ! formatting sequence more efficient.
: 2245 2942 2 !
: 2246 2943 2 !
: 2247 2944 2 CHAR_PTR = CHSFIND NOT CH (.FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT,
: 2248 2945 2 .FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT, CHAR_POUND);
: 2249 2946 2
: 2250 2947 2 IF CH$FAIL (.CHAR_PTR)
: 2251 2948 2 THEN
: 2252 2949 2     BEGIN
: 2253 2950 2     IF .INTEGER_MODE
: 2254 2951 2     THEN
: 2255 2952 2         PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS]
: 2256 2953 2         + .FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT
: 2257 2954 2
: 2258 2955 2     ELSE
: 2259 2956 2         PARAM_BLK [FRACTION_DIGIT] = .PARAM_BLK [FRACTION_DIGIT]
: 2260 2957 2         + .FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT;
: 2261 2958 2         ! remember that there is already
: 2262 2959 2         ! one fraction digit in the '.#' case
: 2263 2960 2
: 2264 2961 2     CHAR_CNT = .FORMAT_STR [DSC$W_LENGTH];
: 2265 2962 2     END
: 2266 2963 2
: 2267 2964 2 ELSE
: 2268 2965 2     BEGIN
: 2269 2966 2     BIND CHARS = (.CHAR_PTR) : VECTOR [, BYTE];
: 2270 2967 2     TMP_LEN = .CHAR_PTR - (.FORMAT_STR [DSC$A_POINTER] + .CHAR_CNT);
: 2271 2968 2     IF .INTEGER_MODE
: 2272 2969 2     THEN
: 2273 2970 2         PARAM_BLK [INTEGER_DIGITS] = .PARAM_BLK [INTEGER_DIGITS]
: 2274 2971 2         + .TMP_LEN
: 2275 2972 2
: 2276 2973 2     ELSE
: 2277 2974 2         PARAM_BLK [FRACTION_DIGIT] = .PARAM_BLK [FRACTION_DIGIT]
: 2278 2975 2         + .TMP_LEN;
: 2279 2976 2         ! remember that there is already
: 2280 2977 2         ! one fraction digit in the '.#' case
: 2281 2978 2
: 2282 2979 2     CHAR_CNT = .CHAR_CNT + .TMP_LEN;
: 2283 2980 2
: 2284 2981 2     SELECTONEU .CHARS [0] OF
: 2285 2982 2     SET
: 2286 2983 2     [CHAR_COMMA] :
: 2287 2984 2
: 2288 2985 2     !+
: 2289 2986 2     ! Commas should not occur in the fractional part.
: 2290 2987 2     !-
: 2291 2988 2
: 2292 2989 2
: 2293 2990 2     IF .INTEGER_MODE
: 2294 2991 2     THEN
: 2295 2992 2         BEGIN
: 2296 2993 2
: 2297 2994 2     !+
: 2298 2995 2     ! Each comma is picked up separately rather than doing a spanc instruction

```

```

: 2299 2996 4 ! with pound signs and commas since the digit separator on output can
: 2300 2997 4 ! have a length other than 1.
: 2301 2998 4 !
: 2302 2999 4 !
: 2303 3000 4 PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK]
: 2304 3001 4 OR COMMA_MASK;
: 2305 3002 4 PARAM_BLK [INTEGER_DIGITS] =
: 2306 3003 4 .PARAM_BLK [INTEGER_DIGITS]
: 2307 3004 4 + .DIGIT_SEP_DESC [DSC$W_LENGTH];
: 2308 3005 4 CHAR_CNT = .CHAR_CNT + 1;
: 2309 3006 4 SPAN_NUMERIC (K SPAN_INTEGER_NO_E,
: 2310 3007 4 FORMAT_STR, CHAR_CNT, PARAM_BLK)
: 2311 3008 4 END;
: 2312 3009 4
: 2313 3010 4 [CHAR_PERIOD] :
: 2314 3011 4
: 2315 3012 4 !+
: 2316 3013 4 ! Periods should not occur in the fractional part.
: 2317 3014 4 !-
: 2318 3015 4
: 2319 3016 4 IF .INTEGER_MODE
: 2320 3017 4 THEN
: 2321 3018 4 BEGIN
: 2322 3019 4 PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK]
: 2323 3020 4 OR PERIOD_MASK;
: 2324 3021 4 CHAR_CNT = .CHAR_CNT + 1;
: 2325 3022 4 SPAN_NUMERIC (.PERIOD_MODE, FORMAT_STR, CHAR_CNT,
: 2326 3023 4 PARAM_BLK)
: 2327 3024 4 END;
: 2328 3025 4
: 2329 3026 4 [CHAR_MINUS] :
: 2330 3027 4 BEGIN
: 2331 3028 4 CHAR_CNT = .CHAR_CNT + 1;
: 2332 3029 4 PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] OR MINUS_MASK
: 2333 3030 4 END;
: 2334 3031 4
: 2335 3032 4 [CHAR_CARAT] :
: 2336 3033 4 IF .E_FLAG
: 2337 3034 4 THEN
: 2338 3035 4 BEGIN
: 2339 3036 4
: 2340 3037 4 !+
: 2341 3038 4 ! If the next three characters are carats, we should indicate that E format
: 2342 3039 4 ! is desired and move CHAR_CNT to last carat.
: 2343 3040 4 !
: 2344 3041 4 ! For efficiency, we make sure that we have four carats beginning with
: 2345 3042 4 ! the character pointed to by CHAR_PTR which we know is a carat.
: 2346 3043 4 !-
: 2347 3044 4
: 2348 3045 4 IF .FORMAT_STR [DSC$W_LENGTH] - .CHAR_CNT GTRU 3
: 2349 3046 4 AND ..CHAR_PTR EQLU .E_FORMAT
: 2350 3047 4 ! Note that we want .. here since
: 2351 3048 4 ! CHAR_PTR is an address
: 2352 3049 4 ! and we want the actual value
: 2353 3050 4 ! pointed to by that address
: 2354 3051 4 THEN
: 2355 3052 4 BEGIN

```

```

: 2356 3053 5          PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK]
: 2357 3054          OR E_MASK;
: 2358 3055          CHAR_CNT = .CHAR_CNT + 4;
: 2359 3056          END
: 2360 3057
: 2361 3058          END;
: 2362 3059
: 2363 3060
: 2364 3061          [CHAR_LF_ANGLE]:
: 2365 3062          !+
: 2366 3063          ! IF '<CD>' is present, it will be the last thing in the format string.
: 2367 3064          ! Call SPAN_BRACKET to check for it.
: 2368 3065          !-
: 2369 3066          SPAN_BRACKET (CHAR_C, FORMAT_STR, PARAM_BLK, CHAR_CNT);
: 2370 3067
: 2371 3068          TES
: 2372 3069
: 2373 3070          END;
: 2374 3071          RETURN 1
: 2375 3072
: 2376 3073
: 2377 3074          1 END;

```

! End of routine SPAN_NUMERIC

| Address | 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 | Op419 | Op420 | Op421 | Op422 | Op423 | Op424 | Op425 | Op426 | Op427 | Op428 | Op429 | Op430 | Op431 | Op432 | Op433 | Op434 | Op435 | Op436 | Op437 | Op438 | Op439 | Op440 | Op441 | Op442 | Op443 | Op444 | Op445 | Op446 | Op447 | Op448 | Op449 | Op450 | Op451 | Op452 | Op453 | Op454 | Op455 | Op456 | Op457 | Op458 | Op459 | Op460 | Op461 | Op462 | Op463 | Op464 | Op465 | Op466 | Op467 | Op468 | Op469 | Op470 | Op471 | Op472 | Op473 | Op474 | Op475 | Op476 | Op477 | Op478 | Op479 | Op480 | Op481 | Op482 | Op483 | Op484 | Op485 | Op486 | Op487 | Op488 | Op489 | Op490 | Op491 | Op492 | Op493 | Op494 | Op495 | Op496 | Op497 | Op498 | Op499 | Op500 | Op501 | Op502 | Op503 | Op504 | Op505 | Op506 | Op507 | Op508 | Op509 | Op510 | Op511 | Op512 | Op513 | Op514 | Op515 | Op516 | Op517 | Op518 | Op519 | Op520 | Op521 | Op522 | Op523 | Op524 | Op525 | Op526 | Op527 | Op528 | Op529 | Op530 | Op531 | Op532 | Op533 | Op534 | Op535 | Op536 | Op537 | Op538 | Op539 | Op540 | Op541 | Op542 | Op543 | Op544 | Op545 | Op546 | Op547 | Op548 | Op549 | Op550 | Op551 | Op552 | Op553 | Op554 | Op555 | Op556 | Op557 | Op558 | Op559 | Op560 | Op561 | Op562 | Op563 | Op564 | Op565 | Op566 | Op567 | Op568 | Op569 | Op570 | Op571 | Op572 | Op573 | Op574 | Op575 | Op576 | Op577 | Op578 | Op579 | Op580 | Op581 | Op582 | Op583 | Op584 | Op585 | Op586 | Op587 | Op588 | Op589 | Op590 | Op591 | Op592 | Op593 | Op594 | Op595 | Op596 | Op597 | Op598 | Op599 | Op600 | Op601 | Op602 | Op603 | Op604 | Op605 | Op606 | Op607 | Op608 | Op609 | Op610 | Op611 | Op612 | Op613 | Op614 | Op615 | Op616 | Op617 | Op618 | Op619 | Op620 | Op621 | Op622 | Op623 | Op624 | Op625 | Op626 | Op627 | Op628 | Op629 | Op630 | Op631 | Op632 | Op633 | Op634 | Op635 | Op636 | Op637 | Op638 | Op639 | Op640 | Op641 | Op642 | Op643 | Op644 | Op645 | Op646 | Op647 | Op648 | Op649 | Op650 | Op651 | Op652 | Op653 | Op654 | Op655 | Op656 | Op657 | Op658 | Op659 | Op660 | Op661 | Op662 | Op663 | Op664 | Op665 | Op666 | Op667 | Op668 | Op669 | Op670 | Op671 | Op672 | Op673 | Op674 | Op675 | Op676 | Op677 | Op678 | Op679 | Op680 | Op681 | Op682 | Op683 | Op684 | Op685 | Op686 | Op687 | Op688 | Op689 | Op690 | Op691 | Op692 | Op693 | Op694 | Op695 | Op696 | Op697 | Op698 | Op699 | Op700 | Op701 | Op702 | Op703 | Op704 | Op705 | Op706 | Op707 | Op708 | Op709 | Op710 | Op711 | Op712 | Op713 | Op714 | Op715 | Op716 | Op717 | Op718 | Op719 | Op720 | Op721 | Op722 | Op723 | Op724 | Op725 | Op726 | Op727 | Op728 | Op729 | Op730 | Op731 | Op732 | Op733 | Op734 | Op735 | Op736 | Op737 | Op738 | Op739 | Op740 | Op741 | Op742 | Op743 | Op744 | Op745 | Op746 | Op747 | Op748 | Op749 | Op750 | Op751 | Op752 | Op753 | Op754 | Op755 | Op756 | Op757 | Op758 | Op759 | Op760 | Op761 | Op762 | Op763 | Op764 | Op765 | Op766 | Op767 | Op768 | Op769 | Op770 | Op771 | Op772 | Op773 | Op774 | Op775 | Op776 | Op777 | Op778 | Op779 | Op780 | Op781 | Op782 | Op783 | Op784 | Op785 | Op786 | Op787 | Op788 | Op789 | Op790 | Op791 | Op792 | Op793 | Op794 | Op795 | Op796 | Op797 | Op798 | Op799 | Op800 | Op801 | Op802 | Op803 | Op804 | Op805 | Op806 | Op807 | Op808 | Op809 | Op810 | Op811 | Op812 | Op813 | Op814 | Op815 | Op816 | Op817 | Op818 | Op819 | Op820 | Op821 | Op822 | Op823 | Op824 | Op825 | Op826 | Op827 | Op828 | Op829 | Op830 | Op831 | Op832 | Op833 | Op834 | Op835 | Op836 | Op837 | Op838 | Op839 | Op840 | Op841 | Op842 | Op843 | Op844 | Op845 | Op846 | Op847 | Op848 | Op849 | Op850 | Op851 | Op852 | Op853 | Op854 | Op855 | Op856 | Op857 | Op858 | Op859 | Op860 | Op861 | Op862 | Op863 | Op864 | Op865 | Op866 | Op867 | Op868 | Op869 | Op870 | Op871 | Op872 | Op873 | Op874 | Op875 | Op876 | Op877 | Op878 | Op879 | Op880 | Op881 | Op882 | Op883 | Op884 | Op885 | Op886 | Op887 | Op888 | Op889 | Op890 | Op891 | Op892 | Op893 | Op894 | Op895 | Op896 | Op897 | Op898 | Op899 | Op900 | Op901 | Op902 | Op903 | Op904 | Op905 | Op906 | Op907 | Op908 | Op909 | Op910 | Op911 | Op912 | Op913 | Op914 | Op915 | Op916 | Op917 | Op918 | Op919 | Op920 | Op921 | Op922 | Op923 | Op924 | Op925 | Op926 | Op927 | Op928 | Op929 | Op930 | Op931 | Op932 | Op933 | Op934 | Op935 | Op936 | Op937 | Op938 | Op939 | Op940 | Op941 | Op942 | Op943 | Op944 | Op945 | Op946 | Op947 | Op948 | Op949 | Op950 | Op951 | Op952 | Op953 | Op954 | Op955 | Op956 | Op957 | Op958 | Op959 | Op960 | Op961 | Op962 | Op963 | Op964 | Op965 | Op966 | Op967 | Op968 | Op969 | Op970 | Op971 | Op972 | Op973 | Op974 | Op975 | Op976 | Op977 | Op978 | Op979 | Op980 | Op981 | Op982 | Op983 | Op984 | Op985 | Op986 | Op987 | Op988 | Op989 | Op990 | Op991 | Op992 | Op993 | Op994 | Op995 | Op996 | Op997 | Op998 | Op999 | Op1000 |
|---------|----|------|------|------|-----|-----|-------|------|-------|------|--------|---------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
| 0022 | 03 | 001D | 0013 | 08 | AC | D0 | 00002 | 000B | 00013 | 1\$: | .WORD | Save R2,R3,R4,R5,R6,R7,R8 | 2826 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | 10 | AC | D0 | 00006 | | | | MOV L | P_FORMAT_STR, R4 | 2883 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | 0C | AC | D0 | 0000A | | | | MOV L | P_PARAM_BLK, R2 | 2884 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | 04 | AC | CF | 0000E | | | | CASE L | SPAN_MODE, #1, #3 | 2885 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | .WORD | 2\$-1\$,- | 2897 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | 3\$-1\$,- | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | 5\$-1\$,- | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | 6\$-1\$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | 22\$ | 2932 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 56 | 00F1 | 31 | D0 | 0001B | | | | BR W | 2\$ | 2902 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 58 | | 01 | D0 | 0001E | | | | MOV L | #1, E_FLAG | 2903 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 03 | D0 | 00021 | | | | MOV L | #3, PERIOD_MODE | 2904 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 05 | 11 | 00024 | | | | BR B | 4\$ | 2909 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 58 | | 56 | D4 | 00026 | | | | CL RL | E_FLAG | 2910 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 57 | | 04 | D0 | 00028 | | | | MOV L | #2, PERIOD_MODE | 2911 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 01 | D0 | 0002B | | | | MOV L | #1, INTEGER_MODE | 2916 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | 09 | 11 | 0002E | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | |
|----|----|-----------|----|----|----|-------|-------|--------|-----------------------|-----------------|------|
| | | 0E | | 57 | E9 | 00050 | | BLBC | INTEGER MODE, 10\$ | 2950 | |
| | | 50 | | 64 | 3C | 00053 | | MOVZWL | (R4), R0 | 2953 | |
| 14 | A2 | 50 | 14 | A2 | C0 | 00056 | | ADDL2 | 20(R2), R0 | | |
| | | 50 | | 63 | C3 | 0005A | | SUBL3 | (R3), R0, 20(R2) | | |
| | | | | 0C | 11 | 0005F | | BRB | 11\$ | 2952 | |
| | | 50 | | 64 | 3C | 00061 | 10\$: | MOVZWL | (R4), R0 | 2957 | |
| | | 50 | 10 | A2 | C0 | 00064 | | ADDL2 | 16(R2), R0 | | |
| 10 | A2 | 50 | | 63 | C3 | 00068 | | SUBL3 | (R3), R0, 16(R2) | | |
| | | 63 | | 64 | 3C | 0006D | 11\$: | MOVZWL | (R4), (R3) | 2961 | |
| | | | | 60 | 11 | 00070 | | BRB | 18\$ | 2947 | |
| | | 51 | | 55 | C3 | 00072 | 12\$: | SUBL3 | R5, CHAR_PTR, TMP_LEN | 2967 | |
| | | 06 | | 57 | E9 | 00076 | | BLBC | INTEGER MODE, 13\$ | 2971 | |
| | | 14 | A2 | 50 | C0 | 00079 | | ADDL2 | TMP_LEN, 20(R2) | | |
| | | | | 04 | 11 | 0007D | | BRB | 14\$ | 2970 | |
| | | 10 | A2 | 50 | C0 | 0007F | 13\$: | ADDL2 | TMP_LEN, 16(R2) | 2975 | |
| | | | | 63 | 50 | C0 | 00083 | 14\$: | ADDL2 | TMP_LEN, (R3) | 2979 |
| | | | | 2C | 61 | 91 | 00086 | | CMPB | (CHAR_PTR), #44 | 2984 |
| | | | | | 1E | 12 | 00089 | | BNEQ | 15\$ | |
| | | 7D | | 57 | E9 | 0008B | | BLBC | INTEGER MODE, 21\$ | 2990 | |
| 04 | A2 | | | 04 | 88 | 0008E | | BISB2 | #4, 4(R2) | 3001 | |
| | 50 | 00000000' | | FF | 3C | 00092 | | MOVZWL | DIGIT_SEP_DESC, R0 | 3004 | |
| | 14 | A2 | | 50 | C0 | 00099 | | ADDL2 | R0, 20(R2) | | |
| | | | | 63 | D6 | 0009D | | INCL | (R3) | 3005 | |
| | | | | 52 | DD | 0009F | | PUSHL | R2 | 3006 | |
| | | | | 53 | DD | 000A1 | | PUSHL | R3 | | |
| | | | | 54 | DD | 000A3 | | PUSHL | R4 | | |
| | | | | 02 | DD | 000A5 | | PUSHL | #2 | | |
| | | | | 17 | 11 | 000A7 | | BRB | 16\$ | | |
| | | 2E | | 61 | 91 | 000A9 | 15\$: | CMPB | (CHAR_PTR), #46 | 3010 | |
| | | | | 19 | 12 | 000AC | | BNEQ | 17\$ | | |
| | | 04 | 5A | 57 | E9 | 000AE | | BLBC | INTEGER MODE, 21\$ | 3016 | |
| | | | A2 | 40 | 8F | 88 | 000B1 | | BISB2 | #64, 4(R2) | 3020 |
| | | | | 63 | D6 | 000B6 | | INCL | (R3) | 3021 | |
| | | | | 52 | DD | 000B8 | | PUSHL | R2 | 3022 | |
| | | | | 53 | DD | 000BA | | PUSHL | R3 | | |
| | | | | 54 | DD | 000BC | | PUSHL | R4 | | |
| | | | | 58 | DD | 000BE | | PUSHL | R4 | | |
| | | FF3B | CF | 04 | FB | 000C0 | 16\$: | CALLS | #4, SPAN_NUMERIC | | |
| | | | | 44 | 11 | 000C5 | | BRB | 21\$ | 3016 | |
| | | | 2D | 61 | 91 | 000C7 | 17\$: | CMPB | (CHAR_PTR), #45 | 3026 | |
| | | | | 08 | 12 | 000CA | | BNEQ | 19\$ | | |
| | | | | 63 | D6 | 000CC | | INCL | (R3) | 3028 | |
| | | 04 | A2 | 02 | 88 | 000CE | | BISB2 | #2, 4(R2) | 3029 | |
| | | | | 37 | 11 | 000D2 | 18\$: | BRB | 21\$ | | |
| | | 5E | 8F | 61 | 91 | 000D4 | 19\$: | CMPB | (CHAR_PTR), #94 | 3032 | |
| | | | | 1F | 12 | 000D8 | | BNEQ | 20\$ | | |
| | | | 2E | 56 | E9 | 000DA | | BLBC | E_FLAG, 21\$ | 3033 | |
| | | | 50 | 64 | 3C | 000DD | | MOVZWL | (R4), R0 | 3045 | |
| | | | 50 | 63 | C2 | 000E0 | | SUBL2 | (R3), R0 | | |
| | | | 03 | 50 | D1 | 000E3 | | CMPL | R0, #3 | | |
| | | | | 23 | 1B | 000E6 | | BLEQU | 21\$ | | |
| | | F6DA | CF | 61 | D1 | 000E8 | | CMPL | (CHAR_PTR), E_FORMAT | 3046 | |
| | | | | 1C | 12 | 000ED | | BNEQ | 21\$ | | |
| | | 04 | A2 | 80 | 8F | 88 | 000EF | | BISB2 | #128, 4(R2) | 3054 |
| | | | 63 | 04 | C0 | 000F4 | | ADDL2 | #4, (R3) | 3055 | |
| | | | | 12 | 11 | 000F7 | | BRB | 21\$ | 3033 | |
| | | | | 3C | 61 | 91 | 000F9 | 20\$: | CMPB | (CHAR_PTR), #60 | 3061 |

BAS\$\$FOR_INT
2-013

BAS\$\$FORMAT_INT - Basic format interpreter
SPAN_NUMERIC - processing numeric format class

N 9
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 63
(10)

| | | | | | | | | | |
|------|----|----|----|----|-------|-------|--------|------------------|--|
| | | | 0D | 12 | 000FC | | BNEQ | 21\$ | |
| | | | 0C | BB | 000FE | | PUSHR | #*M<R2,R3> | |
| | | | 54 | DD | 00100 | | PUSHL | R4 | |
| | 7E | 43 | 8F | 9A | 00102 | | MOVZBL | #67, -(SP) | |
| FE35 | CF | | 04 | FB | 00106 | | CALLS | #4, SPAN_BRACKET | |
| | 50 | | 01 | D0 | 0010B | 21\$: | MOVL | #1, R0 | |
| | | | | 04 | 0010E | | RET | | |
| | | | 50 | D4 | 0010F | 22\$: | CLRL | R0 | |
| | | | | 04 | 00111 | | RET | | |

.....
3066
.....
3072
.....
3074
.....

; Routine Size: 274 bytes, Routine Base: _BAS\$CODE + 0839

```

: 2379 3075 1 %SBTTL 'OUTPUT_ARG - convert and format'
: 2380 3076 1 ROUTINE OUTPUT_ARG ( ! convert and format the argument
: 2381 3077 1
: 2382 3078 1 P_PARAM_BLK) = ! print using paramter block
: 2383 3079 1
: 2384 3080 2 BEGIN
: 2385 3081 2
: 2386 3082 2 !++
: 2387 3083 2
: 2388 3084 2 FUNCTIONAL DESCRIPTION:
: 2389 3085 2
: 2390 3086 2 This action routine is the heart of the whole thing.
: 2391 3087 2 All of the formatting and most of the error checking will be done here.
: 2392 3088 2
: 2393 3089 2 FORMAL PARAMETERS:
: 2394 3090 2
: 2395 3091 2 P_PARAM_BLK.mr.r - parameter block
: 2396 3092 2
: 2397 3093 2 IMPLICIT INPUTS:
: 2398 3094 2
: 2399 3095 2 CURRENCY_DESC - descriptor for currency symbol
: 2400 3096 2 DIGIT_SEP_DESC - descriptor for digit group separator
: 2401 3097 2 RADIX_PT_DESC - descriptor for radix point
: 2402 3098 2
: 2403 3099 2 IMPLICIT OUTPUTS:
: 2404 3100 2
: 2405 3101 2 NONE
: 2406 3102 2
: 2407 3103 2 ROUTINE VALUE:
: 2408 3104 2
: 2409 3105 2 NONE
: 2410 3106 2
: 2411 3107 2 SIDE EFFECTS:
: 2412 3108 2
: 2413 3109 2 NONE
: 2414 3110 2
: 2415 3111 2 !--
: 2416 3112 2
: 2417 3113 2 LOCAL
: 2418 3114 2 TEMP_MASK, ! temp for param_blk [pu_mask]
: 2419 3115 2 DSC := BLOCK [8, BYTE], ! temp for converting and concatenating
: 2420 3116 2 OUT_STR_LEN; ! length returned by conversion routine
: 2421 3117 2
: 2422 3118 2 BIND
: 2423 3119 2 PARAM_BLK = (.P_PARAM_BLK) : BLOCK [, BYTE];
: 2424 3120 2
: 2425 3121 2 !+
: 2426 3122 2 ! Do error checking for the format field.
: 2427 3123 2 !-
: 2428 3124 2
: 2429 3125 2 !+
: 2430 3126 2 ! Set up a temp dynamic string descriptor to use for numeric and text conversions and
: 2431 3127 2 ! string concatenates.
: 2432 3128 2 !-
: 2433 3129 2
: 2434 3130 2 DSC [DSC$W_LENGTH] = 0;
: 2435 3131 2 DSC [DSC$B_CLASS] = DSC$K_CLASS_D;

```

```

: 2436      3132  2   DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2437      3133  2   DSC [DSC$A_POINTER] = 0;
: 2438      3134  2
: 2439      3135  2
: 2440      3136  2   !+
: 2441      3137  2   ! Use a temporary print using mask in the following SELECT statements.
: 2442      3138  2   ! This is because zero_mask and percent_mask fall outside the range
: 2443      3139  2   ! 0 to e_mask, but they are f formats.
: 2444      3140  2   !-
: 2445      3141  2   TEMP_MASK = .PARAM_BLK [PU_MASK];
: 2446      3142  2
: 2447      3143  2   IF (.PARAM_BLK [PU_MASK] AND ZERO_MASK) NEQ 0
: 2448      3144  2   THEN TEMP_MASK = .TEMP_MASK XOR ZERO_MASK;
: 2449      3145  2
: 2450      3146  2   IF (.PARAM_BLK [PU_MASK] AND PERCENT_MASK) NEQ 0
: 2451      3147  2   THEN TEMP_MASK = .TEMP_MASK XOR PERCENT_MASK;
: 2452      3148  2
: 2453      3149  2   IF (.PARAM_BLK [PU_MASK] AND CD_MASK) NEQ 0
: 2454      3150  2   THEN TEMP_MASK = .TEMP_MASK XOR CD_MASK;
: 2455      3151  2   !+
: 2456      3152  2   ! This is where the action is. A case is done on the data type to determine
: 2457      3153  2   ! the proper syntaxing and the conversions which are necessary. The conversion
: 2458      3154  2   ! routines do most of the formatting.
: 2459      3155  2   !-
: 2460      3156  2
: 2461      3157  2   CASE .PARAM_BLK [ELEM_TYPE] FROM DSC$K_DTYPE_B TO DSC$K_DTYPE_H OF
: 2462      3158  2   SET
: 2463      3159  2
: 2464      3160  2   [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L] :
: 2465      3161  2   BEGIN
: 2466      3162  2
: 2467      3163  2   !+
: 2468      3164  2   ! Integers.
: 2469      3165  2   ! Do the necessary syntax checking.
: 2470      3166  2   ! Convert the value to double precision and call the appropriate
: 2471      3167  2   ! conversion.
: 2472      3168  2   !-
: 2473      3169  2
: 2474      3170  2   LOCAL
: 2475      3171  2   D_VALUE : VECTOR [2];           ! hold double precision floating value
: 2476      3172  2
: 2477      3173  2   CVTLD (.PARAM_BLK [ELEM], D_VALUE [0]);
: 2478      3174  2
: 2479      3175  2   SELECTONEU .TEMP_MASK OF
: 2480      3176  2   SET
: 2481      3177  2
: 2482      3178  2   [0 TO E_MASK - 1] :
: 2483      3179  2
: 2484      3180  2   !+
: 2485      3181  2   ! F format
: 2486      3182  2   !-
: 2487      3183  2
: 2488      3184  2   IF NOT BASSCVT_OUT_D F (D_VALUE, .PARAM_BLK [INTEGER_DIGITS],
: 2489      3185  2   .PARAM_BLK [FRACTION_DIGIT],
: 2490      3186  2   (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
: 2491      3187  2   .PARAM_BLK [PU_MASK]),
: 2492      3188  2   OUT_STR_LEN, DSC, 0,           ! Scale factor

```

```

: 2493 3189 3
: 2494 3190 3
: 2495 3191 4
: 2496 3192 4
: 2497 3193 4
: 2498 3194 4
: 2499 3195 4
: 2500 3196 4
: 2501 3197 4
: 2502 3198 4
: 2503 3199 4
: 2504 3200 4
: 2505 3201 4
: 2506 3202 4
: 2507 3203 4
: 2508 3204 4
: 2509 3205 4
: 2510 3206 4
: 2511 3207 4
: 2512 3208 3
: 2513 3209 3
: 2514 3210 3
: 2515 3211 3
: 2516 3212 3
: 2517 3213 3
: 2518 3214 3
: 2519 3215 3
: 2520 3216 3
: 2521 3217 3
: 2522 3218 3
: 2523 3219 4
: 2524 3220 3
: 2525 3221 3
: 2526 3222 3
: 2527 3223 4
: 2528 3224 4
: 2529 3225 4
: 2530 3226 4
: 2531 3227 4
: 2532 3228 4
: 2533 3229 4
: 2534 3230 4
: 2535 3231 4
: 2536 3232 4
: 2537 3233 4
: 2538 3234 4
: 2539 3235 4
: 2540 3236 4
: 2541 3237 4
: 2542 3238 4
: 2543 3239 4
: 2544 3240 3
: 2545 3241 3
: 2546 3242 3
: 2547 3243 3
: 2548 3244 3
: 2549 3245 3

      CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
THEN
  BEGIN
    !+
    ! The number will not fit into the field width supplied. So the
    ! number is returned in Print format with a '%' appended.
    !-

    LOCAL
      PERCENT_DESC : BLOCK [8, BYTE];

    PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
    PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
    PERCENT_DESC [DSC$W_LENGTH] = 1;
    PERCENT_DESC [DSC$A_POINTER] = PERCENT;
    BAS$CVT_OUT_D_G (D_VALUE, 0, OUT_STR_LEN, DSC, 0);
    STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
  END
ELSE
  STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
[ E_MASK TO R_JUSTIFY_MASK - 1 ] :
  !+
  ! E format
  !-

  IF NOT BAS$CVT_OUT_D_E (D_VALUE, .PARAM_BLK [INTEGER_DIGITS],
    .PARAM_BLK [FRACTION_DIGIT],
    (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK]
    ), OUT_STR_LEN, .PARAM_BLK [RET_STR], 0, ! Scale factor
    CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
  THEN
    BEGIN
      !+
      ! The number will not fit into the field width supplied. So the
      ! number is returned in Print format with a '%' appended.
      !-

      LOCAL
        PERCENT_DESC : BLOCK [8, BYTE];

      PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
      PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
      PERCENT_DESC [DSC$W_LENGTH] = 1;
      PERCENT_DESC [DSC$A_POINTER] = PERCENT;
      BAS$CVT_OUT_D_G (D_VALUE, 0, OUT_STR_LEN, DSC, 0);
      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
    END
  ELSE
    STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
[ OTHERWISE ] :
  !+
  ! Text formats
  !-

```

```

: 2550      3246      :-
: 2551      3247      BAS$$STOP (BAS$$PRIUSIFOR);
: 2552      3248      TES;
: 2553      3249
: 2554      3250      END;
: 2555      3251
: 2556      3252      [DSC$$K_DTYPE_F] :
: 2557      3253      !+
: 2558      3254      ! Floating - single precision
: 2559      3255      !-
: 2560      3256
: 2561      3257
: 2562      3258      BEGIN
: 2563      3259
: 2564      3260      SELECTONEU .TEMP_MASK OF
: 2565      3261      SET
: 2566      3262
: 2567      3263      [0 TO E_MASK - 1] :
: 2568      3264
: 2569      3265      !+
: 2570      3266      ! F format
: 2571      3267      !-
: 2572      3268
: 2573      3269      IF NOT BAS$$CVT_OUT_F F (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2574      3270      .PARAM_BLK [FRACTION_DIGIT],
: 2575      3271      (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
: 2576      3272      .PARAM_BLK [PU_MASK]),
: 2577      3273      OUT_STR_LEN, DSC,
: 2578      3274      CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2579      3275
: 2580      3276      THEN
: 2581      3277      BEGIN
: 2582      3278
: 2583      3279      !+
: 2584      3280      ! The number will not fit into the field width supplied. So the
: 2585      3281      ! number is returned in Print format with a '%' appended.
: 2586      3282      !-
: 2587      3283
: 2588      3284      LOCAL
: 2589      3285      PERCENT_DESC : BLOCK [8, BYTE];
: 2590      3286
: 2591      3287      PERCENT_DESC [DSC$$B_CLASS] = DSC$$K_CLASS_S;
: 2592      3288      PERCENT_DESC [DSC$$B_DTYPE] = DSC$$K_DTYPE_T;
: 2593      3289      PERCENT_DESC [DSC$$W_LENGTH] = 1;
: 2594      3290      PERCENT_DESC [DSC$$A_POINTER] = PERCENT;
: 2595      3291      BAS$$CVT_OUT_D G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC, 0);
: 2596      3292      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2597      3293      END
: 2598      3294      ELSE
: 2599      3295      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2600      3296
: 2601      3297      [E_MASK TO R_JUSTIFY_MASK - 1] :
: 2602      3298
: 2603      3299      !+
: 2604      3300      ! E format
: 2605      3301      !-
: 2606      3302
: 2606      3302      IF NOT BAS$$CVT_OUT_F_E (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],

```

```

2607 3303 3
2608 3304 .PARAM_BLK [FRACTION_DIGIT],
2609 3305 (PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK
2610 3306 ), OUT_STR_LEN, .PARAM_BLK [RET_STR],
2611 3307 CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
2612 3308 THEN
2613 3309 !+ E format will only not fit when a negative number is expected to fit in a
2614 3310 !- field of width 1.
2615 3311
2616 3312 BEGIN
2617 3313
2618 3314 !+
2619 3315 !- The number will not fit into the field width supplied. So the
2620 3316 !- number is returned in Print format with a '%' appended.
2621 3317
2622 3318
2623 3319 LOCAL
2624 3320 PERCENT_DESC : BLOCK [8, BYTE];
2625 3321
2626 3322 PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
2627 3323 PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2628 3324 PERCENT_DESC [DSC$W_LENGTH] = 1;
2629 3325 PERCENT_DESC [DSC$A_POINTER] = PERCENT;
2630 3326 BAS$CVT_OUT_D_G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC, 0);
2631 3327 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
2632 3328 END
2633 3329 ELSE
2634 3330 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
2635 3331
2636 3332 [OTHERWISE] :
2637 3333
2638 3334 !+
2639 3335 !- Text formats
2640 3336
2641 3337
2642 3338 BAS$$STOP (BAS$K_PRIUSIFOR);
2643 3339 TES;
2644 3340
2645 3341 END;
2646 3342
2647 3343 [DSC$K_DTYPE_D] :
2648 3344
2649 3345 !+
2650 3346 !- Floating - double precision
2651 3347
2652 3348
2653 3349 BEGIN
2654 3350
2655 3351 SELECTONEU .TEMP_MASK OF
2656 3352 SET
2657 3353
2658 3354 [0 TO E_MASK - 1] :
2659 3355
2660 3356 !+
2661 3357 !- F format
2662 3358
2663 3359

```

```

: 2664      3360      3      IF NOT BASSCVT_OUT_D F (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2665      3361      3      .PARAM_BLK [FRACTION_DIGIT],
: 2666      3362      4      (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
: 2667      3363      4      .PARAM_BLK [PU_MASK]),
: 2668      3364      3      OUT_STR_LEN, DSC, .PARAM_BLK [SCALE_FACTOR],      ! Scale factor
: 2669      3365      3      CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2670      3366      3      THEN
: 2671      3367      4      BEGIN
: 2672      3368      4      !+
: 2673      3369      4      ! The number will not fit into the field width supplied. So the
: 2674      3370      4      ! number is returned in Print format with a '%' appended.
: 2675      3371      4      !-
: 2676      3372      4
: 2677      3373      4      LOCAL
: 2678      3374      4      PERCENT_DESC : BLOCK [8, BYTE];
: 2679      3375      4
: 2680      3376      4      PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 2681      3377      4      PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2682      3378      4      PERCENT_DESC [DSC$W_LENGTH] = 1;
: 2683      3379      4      PERCENT_DESC [DSC$A_POINTER] = PERCENT;
: 2684      3380      4      BASSCVT_OUT_D G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC, .PARAM_BLK [SCALE_FACTOR]);
: 2685      3381      4      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2686      3382      4      END
: 2687      3383      4      ELSE
: 2688      3384      3      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2689      3385      3
: 2690      3386      3      [E_MASK TO R_JUSTIFY_MASK - 1] :
: 2691      3387      3
: 2692      3388      3      !+
: 2693      3389      3      ! E format
: 2694      3390      3      !-
: 2695      3391      3
: 2696      3392      3
: 2697      3393      3      IF NOT BASSCVT_OUT_D E (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2698      3394      3      .PARAM_BLK [FRACTION_DIGIT],
: 2699      3395      4      (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK]
: 2700      3396      3      ), OUT_STR_LEN, .PARAM_BLK [RET_STR], .PARAM_BLK [SCALE_FACTOR],      ! Scale fact
: 2701      3397      3      CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2702      3398      3      THEN
: 2703      3399      4      BEGIN
: 2704      3400      4      !+
: 2705      3401      4      ! The number will not fit into the field width supplied. So the
: 2706      3402      4      ! number is returned in Print format with a '%' appended.
: 2707      3403      4      !-
: 2708      3404      4
: 2709      3405      4      LOCAL
: 2710      3406      4      PERCENT_DESC : BLOCK [8, BYTE];
: 2711      3407      4
: 2712      3408      4      PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 2713      3409      4      PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2714      3410      4      PERCENT_DESC [DSC$W_LENGTH] = 1;
: 2715      3411      4      PERCENT_DESC [DSC$A_POINTER] = PERCENT;
: 2716      3412      4      BASSCVI_OUT_D G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC, .PARAM_BLK [SCALE_FACTOR]);
: 2717      3413      4      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2718      3414      4      END
: 2719      3415      4      ELSE
: 2720      3416      3

```

```

: 2721          3417          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2722          3418
: 2723          3419          [OTHERWISE] :
: 2724          3420
: 2725          3421          | +
: 2726          3422          | | Text formats
: 2727          3423          | -
: 2728          3424
: 2729          3425          BAS$$STOP (BAS$$PRIUSIFOR);
: 2730          3426          TES;
: 2731          3427
: 2732          3428          END;
: 2733          3429
: 2734          3430          [DSC$$DTYPE_P] :
: 2735          3431
: 2736          3432          | +
: 2737          3433          | | Packed decimal string
: 2738          3434          | -
: 2739          3435
: 2740          3436          BEGIN
: 2741          3437
: 2742          3438          SELECTONEU .TEMP_MASK OF
: 2743          3439          SET
: 2744          3440          [0 TO E_MASK - 1] :
: 2745          3441
: 2746          3442          | +
: 2747          3443          | | F format
: 2748          3444          | -
: 2749          3445
: 2750          3446
: 2751          3447          IF NOT BAS$CVT_OUT_P F (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2752          3448          .PARAM_BLK [FRACTION_DIGIT],
: 2753          3449          (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
: 2754          3450          .PARAM_BLK [PU_MASK]),
: 2755          3451          OUT_STR_LEN, DSC,
: 2756          3452          CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2757          3453          THEN
: 2758          3454          BEGIN
: 2759          3455
: 2760          3456          | +
: 2761          3457          | | The number will not fit into the field width supplied. So the
: 2762          3458          | | number is returned in Print format with a '%' appended.
: 2763          3459          | -
: 2764          3460
: 2765          3461          LOCAL
: 2766          3462          PERCENT_DESC : BLOCK [8, BYTE];
: 2767          3463
: 2768          3464          PERCENT_DESC [DSC$$CLASS] = DSC$$CLASS_S;
: 2769          3465          PERCENT_DESC [DSC$$DTYPE] = DSC$$DTYPE_T;
: 2770          3466          PERCENT_DESC [DSC$$LENGTH] = 1;
: 2771          3467          PERCENT_DESC [DSC$$POINTER] = PERCENT;
: 2772          3468          BAS$CVT_OUT_P G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
: 2773          3469          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2774          3470          END
: 2775          3471          ELSE
: 2776          3472          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2777          3473

```

```

2778      3474      [E_MASK TO R_JUSTIFY_MASK - 1] :
2779      3475
2780      3476      |
2781      3477      | + E format
2782      3478      | -
2783      3479
2784      3480      IF NOT BASSCVT_OUT_P E (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
2785      3481      .PARAM_BLK [FRACTION_DIGIT],
2786      3482      (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK
2787      3483      ), OUT_STR_LEN, .PARAM_BLK [RET_STR],
2788      3484      CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
2789      3485      THEN
2790      3486
2791      3487      | + E format will only not fit when a negative number is expected to fit in a
2792      3488      | field of width 1.
2793      3489      | -
2794      3490      BEGIN
2795      3491
2796      3492      | +
2797      3493      | The number will not fit into the field width supplied. So the
2798      3494      | number is returned in Print format with a '%' appended.
2799      3495      | -
2800      3496
2801      3497      LOCAL
2802      3498      PERCENT_DESC : BLOCK [8, BYTE];
2803      3499
2804      3500      PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
2805      3501      PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2806      3502      PERCENT_DESC [DSC$W_LENGTH] = 1;
2807      3503      PERCENT_DESC [DSC$A_POINTER] = PERCENT;
2808      3504      BASSCVT_OUT_P G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
2809      3505      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
2810      3506      END
2811      3507      ELSE
2812      3508      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
2813      3509
2814      3510      [OTHERWISE] :
2815      3511
2816      3512      | +
2817      3513      | Text formats
2818      3514      | -
2819      3515
2820      3516      BASS$STOP (BASS$K_PRIUSIFOR);
2821      3517      TES;
2822      3518
2823      3519      END;
2824      3520
2825      3521      [DSC$K_DTYPE_G] :
2826      3522
2827      3523      | +
2828      3524      | G Floating
2829      3525      | -
2830      3526
2831      3527      BEGIN
2832      3528
2833      3529      SELECT ONEU .TEMP_MASK OF
2834      3530      SET

```

```

2835 3531 4
2836 3532 4
2837 3533 4
2838 3534 4
2839 3535 4
2840 3536 4
2841 3537 4
2842 3538 4
2843 3539 4
2844 3540 4
2845 3541 4
2846 3542 4
2847 3543 4
2848 3544 4
2849 3545 4
2850 3546 4
2851 3547 4
2852 3548 4
2853 3549 4
2854 3550 4
2855 3551 4
2856 3552 4
2857 3553 4
2858 3554 4
2859 3555 4
2860 3556 4
2861 3557 4
2862 3558 4
2863 3559 4
2864 3560 4
2865 3561 4
2866 3562 4
2867 3563 4
2868 3564 4
2869 3565 4
2870 3566 4
2871 3567 4
2872 3568 4
2873 3569 4
2874 3570 4
2875 3571 4
2876 3572 4
2877 3573 4
2878 3574 4
2879 3575 4
2880 3576 4
2881 3577 4
2882 3578 4
2883 3579 4
2884 3580 4
2885 3581 4
2886 3582 4
2887 3583 4
2888 3584 4
2889 3585 4
2890 3586 4
2891 3587 4

```

```

[O TO E_MASK - 1] :
  |
  |+ F format
  |-
  IF NOT BASSCVT_OUT_G F (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
    .PARAM_BLK [FRACTION_DIGIT],
    (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
    .PARAM_BLK [PU_MASK]),
    OUT_STR_LEN, DSC
    CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
  THEN
    BEGIN
      |
      |+ The number will not fit into the field width supplied. So the
      | number is returned in Print format with a '%' appended.
      |-
      LOCAL
        PERCENT_DESC : BLOCK [8, BYTE];

        PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
        PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
        PERCENT_DESC [DSC$W_LENGTH] = 1;
        PERCENT_DESC [DSC$A_POINTER] = PERCENT;
        BASSCVT_OUT_G G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
        STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
      END
    ELSE
      STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
[E_MASK TO R_JUSTIFY_MASK - 1] :
  |
  |+ E format
  |-
  IF NOT BASSCVT_OUT_G E (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
    .PARAM_BLK [FRACTION_DIGIT],
    (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK]
    ), OUT_STR_LEN, .PARAM_BLK [RET_STR],
    CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
  THEN
    BEGIN
      |
      |+ The number will not fit into the field width supplied. So the
      | number is returned in Print format with a '%' appended.
      |-
      LOCAL
        PERCENT_DESC : BLOCK [8, BYTE];

        PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;

```

```

: 2892 3588 4 PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2893 3589 4 PERCENT_DESC [DSC$W_LENGTH] = 1;
: 2894 3590 4 PERCENT_DESC [DSC$A_POINTER] = PERCENT;
: 2895 3591 4 BAS$CVT_OUT G G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
: 2896 3592 4 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2897 3593 4 END
: 2898 3594 4 ELSE
: 2899 3595 4 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2900 3596 4
: 2901 3597 4 [OTHERWISE] :
: 2902 3598 4
: 2903 3599 4 | +
: 2904 3600 4 | | Text formats
: 2905 3601 4 | -
: 2906 3602 4
: 2907 3603 4 BAS$$STOP (BAS$K_PRIUSIFOR);
: 2908 3604 4 TES;
: 2909 3605 4
: 2910 3606 4 END;
: 2911 3607 4
: 2912 3608 4 [DSC$K_DTYPE_H] :
: 2913 3609 4
: 2914 3610 4 | +
: 2915 3611 4 | | H Floating
: 2916 3612 4 | -
: 2917 3613 4
: 2918 3614 4 BEGIN
: 2919 3615 4
: 2920 3616 4 SELECTONEU .TEMP_MASK OF
: 2921 3617 4 SET
: 2922 3618 4
: 2923 3619 4 [0 TO E_MASK - 1] :
: 2924 3620 4
: 2925 3621 4 | +
: 2926 3622 4 | | F format
: 2927 3623 4 | -
: 2928 3624 4
: 2929 3625 4 IF NOT BAS$CVT_OUT H F (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2930 3626 4 .PARAM_BLK [FRACTION_DIGIT],
: 2931 3627 4 (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS;
: 2932 3628 4 .PARAM_BLK [PU_MASK]),
: 2933 3629 4 OUT_STR_LEN, DSC
: 2934 3630 4 CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2935 3631 4 THEN
: 2936 3632 4 BEGIN
: 2937 3633 4
: 2938 3634 4 | +
: 2939 3635 4 | | The number will not fit into the field width supplied. So the
: 2940 3636 4 | | number is returned in Print format with a '%' appended.
: 2941 3637 4 | -
: 2942 3638 4
: 2943 3639 4 LOCAL
: 2944 3640 4 PERCENT_DESC : BLOCK [8, BYTE];
: 2945 3641 4
: 2946 3642 4 PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 2947 3643 4 PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2948 3644 4 PERCENT_DESC [DSC$W_LENGTH] = 1;

```

```

: 2949      3645 4          PERCENT_DESC [DSC$A_POINTER] = PERCENT;
: 2950      3646 4          BASS$CVT_OUT_H_G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
: 2951      3647 4          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2952      3648 4          END
: 2953      3649 4          ELSE
: 2954      3650 4          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2955      3651 4          [E_MASK TO R_JUSTIFY_MASK - 1] :
: 2956      3652 4          |
: 2957      3653 4          | + E format
: 2958      3654 4          | -
: 2959      3655 4          |
: 2960      3656 4          |
: 2961      3657 4          |
: 2962      3658 4          IF NOT BASS$CVT_OUT_H_E (.PARAM_BLK [ELEM], .PARAM_BLK [INTEGER_DIGITS],
: 2963      3659 4          | .PARAM_BLK [FRACTION_DIGIT],
: 2964      3660 4          | (.PARAM_BLK [PU_MASK] = .PARAM_BLK [PU_MASK] AND K_CVT_FLAGS; .PARAM_BLK [PU_MASK
: 2965      3661 4          | ), OUT_STR_LEN, .PARAM_BLK [RET_STR],
: 2966      3662 4          | CURRENCY_DESC, DIGIT_SEP_DESC, RADIX_PT_DESC)
: 2967      3663 4          THEN
: 2968      3664 4          BEGIN
: 2969      3665 4          |
: 2970      3666 4          | +
: 2971      3667 4          | The number will not fit into the field width supplied. So the
: 2972      3668 4          | number is returned in Print format with a '%' appended.
: 2973      3669 4          | -
: 2974      3670 4          |
: 2975      3671 4          LOCAL
: 2976      3672 4          PERCENT_DESC : BLOCK [8, BYTE];
: 2977      3673 4          |
: 2978      3674 4          PERCENT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 2979      3675 4          PERCENT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 2980      3676 4          PERCENT_DESC [DSC$W_LENGTH] = 1;
: 2981      3677 4          PERCENT_DESC [DSC$A_POINTER] = PERCENT;
: 2982      3678 4          BASS$CVT_OUT_H_G (.PARAM_BLK [ELEM], 0, OUT_STR_LEN, DSC);
: 2983      3679 4          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], PERCENT_DESC, DSC);
: 2984      3680 4          END
: 2985      3681 4          ELSE
: 2986      3682 4          STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
: 2987      3683 4          |
: 2988      3684 4          [OTHERWISE] :
: 2989      3685 4          |
: 2990      3686 4          | +
: 2991      3687 4          | Text formats
: 2992      3688 4          | -
: 2993      3689 4          |
: 2994      3690 4          BASS$STOP (BASS$K_PRIUSIFOR);
: 2995      3691 4          TES;
: 2996      3692 4          |
: 2997      3693 4          END;
: 2998      3694 4          [DSC$K_DTYPE_T] :
: 2999      3695 4          |
: 3000      3696 4          | +
: 3001      3697 4          | Text
: 3002      3698 4          | -
: 3003      3699 4          |
: 3004      3700 4          |
: 3005      3701 4          BEGIN

```

```

3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062

```

```

3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758

```

```

!+
!- Allocate a temporary and make it a fixed length string so that
!- truncating and padding occur properly.

STR$GET1 DX (PARAM_BLK [CHARACTER], DSC);
DSC [DSC$B_CLASS] = DSC$K_CLASS_S;

!+
!- Based on the bits in PU_MASK, go do the right type of justification

SELECTONEU .PARAM_BLK [PU_MASK] OF
SET
  [L_JUSTIFY_MASK] :
  BEGIN
    !+
    !- Left justify the string in the return string
    !- This should just amount in a copy. Truncate on the right if necessary.

    STR$COPY DX R8 (DSC, .PARAM_BLK [ELEM]);
    STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
    END;

  [R_JUSTIFY_MASK] :
  !+
  !- Right justify the string.
  !- Use concatenate to prefix the right number of spaces to the
  !- front. If overflow occurs, then left justify the string
  !- and truncate on the right.

  BEGIN
  LOCAL
    LENGTH; ! Length of the element to format
  LENGTH = .(.PARAM_BLK [ELEM])<0, 16>;
  IF .LENGTH GEQ .PARAM_BLK [CHARACTER]
  THEN
  BEGIN
    !+
    !- String overflow

    STR$COPY DX R8 (DSC, .PARAM_BLK [ELEM]);
    STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
    END
  ELSE
  BEGIN

```

```

3063 3759 S
3064 3760 S
3065 3761 S
3066 3762 S
3067 3763 S
3068 3764 S
3069 3765 S
3070 3766 4
3071 3767 4
3072 3768 3
3073 3769 3
3074 3770 3
3075 3771 4
3076 3772 4
3077 3773 4
3078 3774 4
3079 3775 4
3080 3776 4
3081 3777 4
3082 3778 4
3083 3779 4
3084 3780 4
3085 3781 4
3086 3782 4
3087 3783 4
3088 3784 4
3089 3785 4
3090 3786 4
3091 3787 5
3092 3788 5
3093 3789 5
3094 3790 5
3095 3791 5
3096 3792 5
3097 3793 5
3098 3794 5
3099 3795 5
3100 3796 4
3101 3797 5
3102 3798 5
3103 3799 5
3104 3800 5
3105 3801 5
3106 3802 5
3107 3803 5
3108 3804 5
3109 3805 5
3110 3806 5
3111 3807 5
3112 3808 5
3113 3809 5
3114 3810 5
3115 3811 5
3116 3812 5
3117 3813 5
3118 3814 5
3119 3815 5

```

```

!+
!- The string will fit; do an RSET.
!-

BASSRSET (DSC, .PARAM_BLK [ELEM]);
STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
END;

END;

[C_JUSTIFY_MASK] :
BEGIN
!+
!- This is a combination of concatenating spaces on the front and
!- the end if necessary.
!- If overflow occurs, left justify the string and truncate on the
!- right.
!-

LOCAL
LENGTH; ! Length of the element to format
LENGTH = .(.PARAM_BLK [ELEM])<0, 16>;
IF .LENGTH GEQ .PARAM_BLK [CHARACTER]
THEN
BEGIN
!+
!- String overflow
!-

STR$COPY_DX_R8 (DSC, .PARAM_BLK [ELEM]);
STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
END
ELSE
BEGIN
!+
!- The string will fit; do an RSET into the left part followed by a copy
!- into the rest of the string
!-

LOCAL
T_LENGTH, ! Temp length for STR$GET1
T_DSC : BLOCK [8, BYTE]; ! Temp descriptor

T_LENGTH = .PARAM_BLK [CHARACTER] - ((.PARAM_BLK [CHARACTER] + 1) - .LENGTH)/2;
T_DSC [DSC$W_LENGTH] = 0;
T_DSC [DSC$B_CLASS] = DSC$K_CLASS_D;
T_DSC [DSC$A_POINTER] = 0;
STR$GET1_DX (T_LENGTH, T_DSC);
T_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
BASSRSET (T_DSC, .PARAM_BLK [ELEM]);
STR$COPY_DX_R8 (DSC, T_DSC);

```

```

3120 3816 5 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
3121 3817 5 T_DSC [DSC$B_CLASS] = DSC$K_CLASS_D;
3122 3818 5 STR$FREE1_DX~(T_DSC);
3123 3819 4 END;
3124 3820 4
3125 3821 4
3126 3822 4
3127 3823 4
3128 3824 4 [EXTEND MASK] :
3129 3825 4 BEGIN
3130 3826 4
3131 3827 4 !+
3132 3828 4 Extended string
3133 3829 4 Copy the text into the destination string and pad with spaces if
3134 3830 4 there is still room at the end.
3135 3831 4
3136 3832 4 LOCAL
3137 3833 4 LENGTH; ! Length of the element to format
3138 3834 4
3139 3835 4 LENGTH = .(.PARAM_BLK [ELEM])<0, 16>;
3140 3836 4
3141 3837 4 IF .LENGTH GEQ .PARAM_BLK [CHARACTER]
3142 3838 4 THEN
3143 3839 4
3144 3840 4 !+
3145 3841 4 String overflow
3146 3842 4
3147 3843 4
3148 3844 4 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], .PARAM_BLK [ELEM])
3149 3845 4 ELSE
3150 3846 4 BEGIN
3151 3847 4
3152 3848 4 !+
3153 3849 4 The string will fit; do a copy.
3154 3850 4
3155 3851 4
3156 3852 4 STR$COPY DX_R8 (DSC, .PARAM_BLK [ELEM]);
3157 3853 4 STR$CONCAT (.PARAM_BLK [RET_STR], .PARAM_BLK [RET_STR], DSC);
3158 3854 4 END;
3159 3855 4
3160 3856 4 END;
3161 3857 4
3162 3858 4 [OTHERWISE] :
3163 3859 4
3164 3860 4 !+
3165 3861 4 Error - format is for numeric element
3166 3862 4
3167 3863 4
3168 3864 4 BAS$$STOP (BAS$K_PRIUSIFOR);
3169 3865 4 TES;
3170 3866 4
3171 3867 4 DSC [DSC$B_CLASS] = DSC$K_CLASS_D;
3172 3868 4 END;
3173 3869 4
3174 3870 4 [INRANGE, OTRANGE] :
3175 3871 4
3176 3872 4 !+

```


| | | | | | | | | | | | |
|-----------|----|-----------|------|----|-------|-------|--------|-------------------------|--|--|------|
| 10 | AE | OC | 052D | 31 | 00076 | 6\$: | BRW | 55\$ | | | |
| 0000007F | 8F | | BA | 6E | 00079 | | CVTLD | @12(R10), D VALUE | | | 3173 |
| | | | 52 | D1 | 0007E | | CMPL | TEMP_MASK, #127 | | | 3178 |
| | | | 35 | 1A | 00085 | | BGTRU | 7\$ | | | |
| | | 00000000' | EF | 9F | 00087 | | PUSHAB | RADIX_PT_DESC | | | 3184 |
| | | 00000000' | EF | 9F | 0008D | | PUSHAB | DIGIT_SEP_DESC | | | |
| | | 00000000' | EF | 9F | 00093 | | PUSHAB | CURRENCY_DESC | | | |
| | | | 7E | D4 | 00099 | | CLRL | -(SP) | | | |
| | | 28 | AE | 9F | 0009B | | PUSHAB | DSC | | | |
| | | 14 | AE | 9F | 0009E | | PUSHAB | OUT_STR_LEN | | | |
| | 6B | FFFF8FA0 | 8F | CA | 000A1 | | BICL2 | #-28768, (R11) | | | 3186 |
| | | | 6B | DD | 000A8 | | PUSHL | (R11) | | | 3187 |
| | | | 10 | AA | DD | 000AA | PUSHL | 16(R10) | | | 3185 |
| | | | 14 | AA | DD | 000AD | PUSHL | 20(R10) | | | 3184 |
| | | | 34 | AE | 9F | 000B0 | PUSHAB | D VALUE | | | |
| 00000000G | 00 | | 0A | FB | 000B3 | | CALLS | #TO, BAS\$CVT_OUT_D_F | | | |
| | | | 47 | 11 | 000BA | | BRB | 10\$ | | | |
| 00000080 | 8F | | 52 | D1 | 000BC | 7\$: | CMPL | TEMP_MASK, #128 | | | 3211 |
| | | | 03 | 1E | 000C3 | | BGEQU | 9\$ | | | |
| | | | 03D1 | 31 | 000C5 | 8\$: | BRW | 43\$ | | | |
| 000000FF | 8F | | 52 | D1 | 000C8 | 9\$: | CMPL | TEMP_MASK, #255 | | | |
| | | | F4 | 1A | 000CF | | BGTRU | 8\$ | | | |
| | | 00000000' | EF | 9F | 000D1 | | PUSHAB | RADIX_PT_DESC | | | 3217 |
| | | 00000000' | EF | 9F | 000D7 | | PUSHAB | DIGIT_SEP_DESC | | | |
| | | 00000000' | EF | 9F | 000DD | | PUSHAB | CURRENCY_DESC | | | |
| | | | 7E | D4 | 000E3 | | CLRL | -(SP) | | | |
| | | | 6A | DD | 000E5 | | PUSHL | (R10) | | | 3220 |
| | | 14 | AE | 9F | 000E7 | | PUSHAB | OUT_STR_LEN | | | 3217 |
| | 6B | FFFF8FA0 | 8F | CA | 000EA | | BICL2 | #-28768, (R11) | | | 3219 |
| | | | 6B | DD | 000F1 | | PUSHL | (R11) | | | |
| | | | 10 | AA | DD | 000F3 | PUSHL | 16(R10) | | | 3218 |
| | | | 14 | AA | DD | 000F6 | PUSHL | 20(R10) | | | 3217 |
| | | | 34 | AE | 9F | 000F9 | PUSHAB | D VALUE | | | |
| 00000000G | 00 | | 0A | FB | 000FC | | CALLS | #TO, BAS\$CVT_OUT_D_E | | | |
| | | | 50 | E9 | 00103 | 10\$: | BLBC | R0, 11\$ | | | |
| | | | 0380 | 31 | 00106 | | BRW | 42\$ | | | |
| 08 | AE | 010E0001 | 8F | D0 | 00109 | 11\$: | MOVL | #17694721, PERCENT_DESC | | | 3235 |
| OC | AE | F5A4 | CF | 9E | 00111 | | MOVAB | PERCENT, PERCENT_DESC+4 | | | 3236 |
| | | | 7E | D4 | 00117 | | CLRL | -(SP) | | | 3237 |
| | | | 1C | AE | 9F | 00119 | PUSHAB | DSC | | | |
| | | | 08 | AE | 9F | 0011C | PUSHAB | OUT_STR_LEN | | | |
| | | | 7E | D4 | 0011F | | CLRL | -(SP) | | | |
| | | | 20 | AE | 9F | 00121 | PUSHAB | D VALUE | | | |
| 00000000G | 00 | | 05 | FB | 00124 | | CALLS | #5, BAS\$CVT_OUT_D_G | | | |
| | | | 18 | AE | 9F | 0012B | PUSHAB | DSC | | | 3238 |
| | | | OC | AE | 9F | 0012E | PUSHAB | PERCENT_DESC | | | |
| | | | 0348 | 31 | 00131 | | BRW | 41\$ | | | |
| 0000007F | 8F | | 52 | D1 | 00134 | 12\$: | CMPL | TEMP_MASK, #127 | | | 3263 |
| | | | 33 | 1A | 0013B | | BGTRU | 13\$ | | | |
| | | 00000000' | EF | 9F | 0013D | | PUSHAB | RADIX_PT_DESC | | | 3269 |
| | | 00000000' | EF | 9F | 00143 | | PUSHAB | DIGIT_SEP_DESC | | | |
| | | 00000000' | EF | 9F | 00149 | | PUSHAB | CURRENCY_DESC | | | |
| | | | 24 | AE | 9F | 0014F | PUSHAB | DSC | | | |
| | | | 10 | AE | 9F | 00152 | PUSHAB | OUT_STR_LEN | | | |
| | 6B | FFFF8FA0 | 8F | CA | 00155 | | BICL2 | #-28768, (R11) | | | 3271 |
| | | | 6B | DD | 0015C | | PUSHL | (R11) | | | 3272 |
| | | | 10 | AA | DD | 0015E | PUSHL | 16(R10) | | | 3270 |

| | | | | | | |
|-----------|----|-------------|--|--|--|------|
| 00000000G | 00 | 14 0C | AA DD 00161 AA DD 00164 09 FB 00167 45 11 0016E | | PUSHL 20(R10) PUSHL 12(R10) CALLS #9, BAS\$CVT_OUT_F_F BRB 16\$ | 3269 |
| 00000080 | 8F | | 52 D1 00170 13\$: 03 1E 00177 | | CMPL TEMP_MASK, #128 BGEQU 15\$ | 3296 |
| 000000FF | 8F | | 031D 31 00179 14\$: 52 D1 0017C 15\$: F4 1A 00183 | | BRW 43\$ CMPL TEMP_MASK, #255 BGTRU 14\$ | |
| | | 00000000* | EF 9F 00185 | | PUSHAB RADIX_PT_DESC | 3302 |
| | | 00000000* | EF 9F 0018B | | PUSHAB DIGIT_SEP_DESC | |
| | | 00000000* | EF 9F 00191 | | PUSHAB CURRENCY_DESC | |
| | | | 6A DD 00197 | | PUSHL (R10) | 3305 |
| | | 10 | AE 9F 00199 | | PUSHAB OUT_STR_LEN | 3302 |
| | 6B | FFFF8FA0 | 8F CA 0019C | | BICL2 #-28768, (R11) | 3304 |
| | | | 6B DD 001A3 | | PUSHL (R11) | |
| | | 10 | AA DD 001A5 | | PUSHL 16(R10) | 3303 |
| | | 14 | AA DD 001A8 | | PUSHL 20(R10) | 3302 |
| | | 0C | AA DD 001AB | | PUSHL 12(R10) | |
| 00000000G | 00 | | 09 FB 001AE | | CALLS #9, BAS\$CVT_OUT_F_E | |
| | 03 | | 50 E9 001B5 16\$: 02CE 31 001B8 | | BLBC R0, 17\$ BRW 42\$ | |
| | 10 | AE 010E0001 | 8F D0 001BB 17\$: | | MOVL #17694721, PERCENT_DESC | 3324 |
| | 14 | AE F4F2 | CF 9E 001C3 | | MOVAB PERCENT, PERCENT_DESC+4 | 3325 |
| | | | 7E D4 001C9 | | CLRL -(SP) | 3326 |
| | | | 009E 31 001CB | | BRW 24\$ | |
| 0000007F | 8F | | 52 D1 001CE 18\$: 36 1A 001D5 | | CMPL TEMP_MASK, #127 BGTRU 19\$ | 3354 |
| | | 00000000* | EF 9F 001D7 | | PUSHAB RADIX_PT_DESC | 3360 |
| | | 00000000* | EF 9F 001DD | | PUSHAB DIGIT_SEP_DESC | |
| | | 00000000* | EF 9F 001E3 | | PUSHAB CURRENCY_DESC | |
| | | 1C | AA DD 001E9 | | PUSHL 28(R10) | 3364 |
| | | 28 | AE 9F 001EC | | PUSHAB DSC | 3360 |
| | | 14 | AE 9F 001EF | | PUSHAB OUT_STR_LEN | |
| | 6B | FFFF8FA0 | 8F CA 001F2 | | BICL2 #-28768, (R11) | 3362 |
| | | | 6B DD 001F9 | | PUSHL (R11) | 3363 |
| | | 10 | AA DD 001FB | | PUSHL 16(R10) | 3361 |
| | | 14 | AA DD 001FE | | PUSHL 20(R10) | 3360 |
| | | 0C | AA DD 00201 | | PUSHL 12(R10) | |
| 00000000G | 00 | | 0A FB 00204 | | CALLS #10, BAS\$CVT_OUT_D_F | |
| | | | 48 11 0020B | | BRB 22\$ | |
| 00000080 | 8F | | 52 D1 0020D 19\$: 03 1E 00214 | | CMPL TEMP_MASK, #128 BGEQU 21\$ | 3387 |
| | | | 0280 31 00216 20\$: 52 D1 00219 21\$: | | BRW 43\$ CMPL TEMP_MASK, #255 | |
| 000000FF | 8F | | F4 1A 00220 | | BGTRU 20\$ | |
| | | 00000000* | EF 9F 00222 | | PUSHAB RADIX_PT_DESC | 3393 |
| | | 00000000* | EF 9F 00228 | | PUSHAB DIGIT_SEP_DESC | |
| | | 00000000* | EF 9F 0022E | | PUSHAB CURRENCY_DESC | |
| | | 1C | AA DD 00234 | | PUSHL 28(R10) | 3396 |
| | | | 6A DD 00237 | | PUSHL (R10) | |
| | | 14 | AE 9F 00239 | | PUSHAB OUT_STR_LEN | 3393 |
| | 6B | FFFF8FA0 | 8F CA 0023C | | BICL2 #-28768, (R11) | 3395 |
| | | | 6B DD 00243 | | PUSHL (R11) | |
| | | 10 | AA DD 00245 | | PUSHL 16(R10) | 3394 |
| | | 14 | AA DD 00248 | | PUSHL 20(R10) | 3393 |
| | | 0C | AA DD 0024B | | PUSHL 12(R10) | |

| | | | | | | | | |
|-----------|----|-----------|------|----------|-------|-------|------------------------|-------------------------|
| 00000000G | 00 | | 0A | FB | 0024E | CALLS | #10, BASS\$CVT_OUT_D_E | |
| | 03 | | 50 | E9 | 00255 | BLBC | R0, 23\$ | |
| | | | 022E | 31 | 00258 | BRW | 42\$ | |
| 10 | AE | 010E0001 | 8F | DO | 0025B | 23\$: | MOVL | #17694721, PERCENT_DESC |
| 14 | AE | F452 | CF | 9E | 00263 | | MOVAB | PERCENT, PERCENT_DESC+4 |
| | | | 1C | AA | DD | 00269 | PUSHL | 28(R10) |
| | | | 1C | AE | 9F | 0026C | 24\$: | PUSHAB |
| | | | 08 | AE | 9F | 0026F | | PUSHAB |
| | | | 7E | D4 | 00272 | | CLRL | -(SP) |
| | | | 0C | AA | DD | 00274 | | PUSHL |
| 00000000G | 00 | | 05 | FB | 00277 | CALLS | #5, BASS\$CVT_OUT_D_G | |
| | | | 01F5 | 31 | 0027E | BRW | 40\$ | |
| 0000007F | 8F | | 52 | D1 | 00281 | 25\$: | CMPL | TEMP_MASK, #127 |
| | | | 33 | 1A | 00288 | | BGTRU | 26\$ |
| | | 00000000' | EF | 9F | 0028A | | PUSHAB | RADIX_PT_DESC |
| | | 00000000' | EF | 9F | 00290 | | PUSHAB | DIGIT_SEP_DESC |
| | | 00000000' | EF | 9F | 00296 | | PUSHAB | CURRENCY_DESC |
| | | | 24 | AE | 9F | 0029C | | PUSHAB |
| | | | 10 | AE | 9F | 0029F | | PUSHAB |
| | | | 6B | FFFF8FA0 | 8F | CA | 002A2 | BICL2 |
| | | | | 6B | DD | 002A9 | | PUSHL |
| | | | 10 | AA | DD | 002AB | | PUSHL |
| | | | 14 | AA | DD | 002AE | | PUSHL |
| | | | 0C | AA | DD | 002B1 | | PUSHL |
| 00000000G | 00 | | 09 | FB | 002B4 | CALLS | #9, BASS\$CVT_OUT_P_F | |
| | | | 45 | 11 | 002BB | BRB | 29\$ | |
| 00000080 | 8F | | 52 | D1 | 002BD | 26\$: | CMPL | TEMP_MASK, #128 |
| | | | 03 | 1E | 002C4 | | BGEQU | 28\$ |
| | | | 01D0 | 31 | 002C6 | 27\$: | BRW | 43\$ |
| 000000FF | 8F | | 52 | D1 | 002C9 | 28\$: | CMPL | TEMP_MASK, #255 |
| | | | F4 | 1A | 002D0 | | BGTRU | 27\$ |
| | | 00000000' | EF | 9F | 002D2 | | PUSHAB | RADIX_PT_DESC |
| | | 00000000' | EF | 9F | 002D8 | | PUSHAB | DIGIT_SEP_DESC |
| | | 00000000' | EF | 9F | 002DE | | PUSHAB | CURRENCY_DESC |
| | | | 6A | DD | 002E4 | | PUSHL | (R10) |
| | | | 10 | AE | 9F | 002E6 | | PUSHAB |
| | | | 6B | FFFF8FA0 | 8F | CA | 002E9 | BICL2 |
| | | | | 6B | DD | 002F0 | | PUSHL |
| | | | 10 | AA | DD | 002F2 | | PUSHL |
| | | | 14 | AA | DD | 002F5 | | PUSHL |
| | | | 0C | AA | DD | 002F8 | | PUSHL |
| 00000000G | 00 | | 09 | FB | 002FB | CALLS | #9, BASS\$CVT_OUT_P_E | |
| | | | 50 | E9 | 00302 | 29\$: | BLBC | R0, 30\$ |
| | | | 0181 | 31 | 00305 | | BRW | 42\$ |
| 10 | AE | 010E0001 | 8F | DO | 00308 | 30\$: | MOVL | #17694721, PERCENT_DESC |
| 14 | AE | F3A5 | CF | 9E | 00310 | | MOVAB | PERCENT, PERCENT_DESC+4 |
| | | | 18 | AE | 9F | 00316 | | PUSHAB |
| | | | 04 | AE | 9F | 00319 | | PUSHAB |
| | | | 7E | D4 | 0031C | | CLRL | -(SP) |
| | | | 0C | AA | DD | 0031E | | PUSHL |
| 00000000G | 00 | | 04 | FB | 00321 | CALLS | #4, BASS\$CVT_OUT_P_G | |
| | | | 014B | 31 | 00328 | BRW | 40\$ | |
| 0000007F | 8F | | 52 | D1 | 0032B | 31\$: | CMPL | TEMP_MASK, #127 |
| | | | 33 | 1A | 00332 | | BGTRU | 32\$ |
| | | 00000000' | EF | 9F | 00334 | | PUSHAB | RADIX_PT_DESC |
| | | 00000000' | EF | 9F | 0033A | | PUSHAB | DIGIT_SEP_DESC |
| | | 00000000' | EF | 9F | 00340 | | PUSHAB | CURRENCY_DESC |

| | | | | | | | | | |
|----------|----|-------------|------|----|-------|--------|-------------------------|-------------------------|------|
| | | 24 | AE | 9F | 00346 | PUSHAB | DSC | | |
| | | 10 | AE | 9F | 00349 | PUSHAB | OUT_STR_LEN | | |
| | 6B | FFFF8FA0 | 8F | CA | 0034C | BICL2 | #-28768, (R11) | | 3540 |
| | | | 6B | DD | 00353 | PUSHL | (R11) | | 3541 |
| | | | 10 | AA | DD | 00355 | PUSHL | 16(R10) | 3539 |
| | | | 14 | AA | DD | 00358 | PUSHL | 20(R10) | 3538 |
| | | 0C | AA | DD | 0035B | PUSHL | 12(R10) | | |
| 0000000G | 00 | | 09 | FB | 0035E | CALLS | #9, BASS\$CVT_OUT_G_F | | |
| | | | 45 | 11 | 00365 | BRB | 35\$ | | |
| 00000080 | 8F | | 52 | D1 | 00367 | 32\$: | CMPL | TEMP_MASK, #128 | 3565 |
| | | | 03 | 1E | 0036E | BGEQU | 34\$ | | |
| | | | 0126 | 31 | 00370 | 33\$: | BRW | 43\$ | |
| 000000FF | 8F | | 52 | D1 | 00373 | 34\$: | CMPL | TEMP_MASK, #255 | |
| | | | F4 | 1A | 0037A | BGTRU | 33\$ | | |
| | | 00000000' | EF | 9F | 0037C | PUSHAB | RADIX_PT_DESC | | 3571 |
| | | 00000000' | EF | 9F | 00382 | PUSHAB | DIGIT_SEP_DESC | | |
| | | 00000000' | EF | 9F | 00388 | PUSHAB | CURRENCY_DESC | | |
| | | | 6A | DD | 0038E | PUSHL | (R10) | | 3574 |
| | | | 10 | AE | 9F | 00390 | PUSHAB | OUT_STR_LEN | 3571 |
| | 6B | FFFF8FA0 | 8F | CA | 00393 | BICL2 | #-28768, (R11) | | 3573 |
| | | | 6B | DD | 0039A | PUSHL | (R11) | | |
| | | | 10 | AA | DD | 0039C | PUSHL | 16(R10) | 3572 |
| | | | 14 | AA | DD | 0039F | PUSHL | 20(R10) | 3571 |
| | | 0C | AA | DD | 003A2 | PUSHL | 12(R10) | | |
| 0000000G | 00 | | 09 | FB | 003A5 | CALLS | #9, BASS\$CVT_OUT_G_E | | |
| | | | 50 | E9 | 003AC | 35\$: | BLBC | R0, 36\$ | |
| | | | 00D7 | 31 | 003AF | BRW | 42\$ | | |
| | 10 | AE 010E0001 | 8F | D0 | 003B2 | 36\$: | MOVL | #17694721, PERCENT_DESC | 3589 |
| | 14 | AE F2FB | CF | 9E | 003BA | MOVAB | PERCENT, PERCENT_DESC+4 | | 3590 |
| | | | 18 | AE | 9F | 003C0 | PUSHAB | DSC | 3591 |
| | | | 04 | AE | 9F | 003C3 | PUSHAB | OUT_STR_LEN | |
| | | | 7E | D4 | 003C6 | CLRL | -(SP) | | |
| | | 0C | AA | DD | 003C8 | PUSHL | 12(R10) | | |
| 0000000G | 00 | | 04 | FB | 003CB | CALLS | #4, BASS\$CVT_OUT_G_G | | |
| | | | 00A1 | 31 | 003D2 | BRW | 40\$ | | 3592 |
| 0000007F | 8F | | 52 | D1 | 003D5 | 37\$: | CMPL | TEMP_MASK, #127 | 3619 |
| | | | 33 | 1A | 003DC | BGTRU | 38\$ | | |
| | | 00000000' | EF | 9F | 003DE | PUSHAB | RADIX_PT_DESC | | 3625 |
| | | 00000000' | EF | 9F | 003E4 | PUSHAB | DIGIT_SEP_DESC | | |
| | | 00000000' | EF | 9F | 003EA | PUSHAB | CURRENCY_DESC | | |
| | | | 24 | AE | 9F | 003F0 | PUSHAB | DSC | |
| | | | 10 | AE | 9F | 003F3 | PUSHAB | OUT_STR_LEN | |
| | 6B | FFFF8FA0 | 8F | CA | 003F6 | BICL2 | #-28768, (R11) | | 3627 |
| | | | 6B | DD | 003FD | PUSHL | (R11) | | 3628 |
| | | | 10 | AA | DD | 003FF | PUSHL | 16(R10) | 3626 |
| | | | 14 | AA | DD | 00402 | PUSHL | 20(R10) | 3625 |
| | | 0C | AA | DD | 00405 | PUSHL | 12(R10) | | |
| 0000000G | 00 | | 09 | FB | 00408 | CALLS | #9, BASS\$CVT_OUT_H_F | | |
| | | | 42 | 11 | 0040F | BRB | 39\$ | | |
| 00000080 | 8F | | 52 | D1 | 00411 | 38\$: | CMPL | TEMP_MASK, #128 | 3652 |
| | | | 7F | 1F | 00418 | BLSSU | 43\$ | | |
| 000000FF | 8F | | 52 | D1 | 0041A | CMPL | TEMP_MASK, #255 | | |
| | | | 76 | 1A | 00421 | BGTRU | 43\$ | | |
| | | 00000000' | EF | 9F | 00423 | PUSHAB | RADIX_PT_DESC | | 3658 |
| | | 00000000' | EF | 9F | 00429 | PUSHAB | DIGIT_SEP_DESC | | |
| | | 00000000' | EF | 9F | 0042F | PUSHAB | CURRENCY_DESC | | |
| | | | 6A | DD | 00435 | PUSHL | (R10) | | 3661 |

| | | | | | | | | | |
|-----------|----|-----------|----------|----|-------|--------|------------------------|-------------------------|------|
| | | 10 | AE | 9F | 00437 | PUSHAB | OUT_STR_LEN | 3658 | |
| | 6B | FFFF8FA0 | 8F | CA | 0043A | BICL2 | #-28768, (R11) | 3660 | |
| | | | 6B | DD | 00441 | PUSHL | (R11) | | |
| | | 10 | AA | DD | 00443 | PUSHL | 16(R10) | 3659 | |
| | | 14 | AA | DD | 00446 | PUSHL | 20(R10) | 3658 | |
| | | 0C | AA | DD | 00449 | PUSHL | 12(R10) | | |
| 00000000G | 00 | | 09 | FB | 0044C | CALLS | #9, BASSCVT_OUT_H_E | | |
| | 33 | | 50 | EB | 00453 | BLBS | R0, 42\$ | | |
| | 10 | AE | 010E0001 | 8F | DD | 00456 | MOVL | #17694721, PERCENT_DESC | 3676 |
| | 14 | AE | F257 | CF | 9E | 0045E | MOVAB | PERCENT, PERCENT_DESC+4 | 3677 |
| | | 18 | AE | 9F | 00464 | PUSHAB | DSC | 3678 | |
| | | 04 | AE | 9F | 00467 | PUSHAB | OUT_STR_LEN | | |
| | | | 7E | D4 | 0046A | CLRL | -(SP) | | |
| | | 0C | AA | DD | 0046C | PUSHL | 12(R10) | | |
| 00000000G | 00 | | 04 | FB | 0046F | CALLS | #4, BASSCVT_OUT_H_G | | |
| | | 18 | AE | 9F | 00476 | PUSHAB | DSC | 3679 | |
| | | 14 | AE | 9F | 00479 | PUSHAB | PERCENT_DESC | | |
| | | | 6A | DD | 0047C | PUSHL | (R10) | | |
| | | | 6A | DD | 0047E | PUSHL | (R10) | | |
| 00000000G | 00 | | 04 | FB | 00480 | CALLS | #4, STR\$CONCAT | | |
| | | | 1D | 11 | 00487 | BRB | 44\$ | 3658 | |
| | | 18 | AE | 9F | 00489 | PUSHAB | DSC | 3682 | |
| | | | 6A | DD | 0048C | PUSHL | (R10) | | |
| | | | 6A | DD | 0048E | PUSHL | (R10) | | |
| 00000000G | 00 | | 03 | FB | 00490 | CALLS | #3, STR\$CONCAT | | |
| | | | 0D | 11 | 00497 | BRB | 44\$ | 3658 | |
| 00000000G | 00 | 00000000G | 8F | DD | 00499 | PUSHL | #BASSK PRIUSIFOR | 3690 | |
| | | | 01 | FB | 0049F | CALLS | #1, BASS\$STOP | | |
| | | | 00FD | 31 | 004A6 | BRW | 55\$ | 3157 | |
| | | 18 | AE | 9F | 004A9 | PUSHAB | DSC | 3708 | |
| | | 18 | AA | 9F | 004AC | PUSHAB | 24(R10) | | |
| 00000000G | 00 | | 02 | FB | 004AF | CALLS | #2, STR\$GET1_DX | | |
| | 18 | AE | 01 | 90 | 004B6 | MOVB | #1, DSC+3 | 3709 | |
| 00000800 | 8F | | 6B | D1 | 004BA | CMPL | (R11), #2048 | 3718 | |
| | | | 03 | 12 | 004C1 | BNEQ | 47\$ | | |
| | | | 00B1 | 31 | 004C3 | BRW | 50\$ | | |
| 00000100 | 8F | | 6B | D1 | 004C6 | CMPL | (R11), #256 | 3730 | |
| | | | 1A | 12 | 004CD | BNEQ | 48\$ | | |
| | | | BA | 3C | 004CF | MOVZWL | @12(R10), LENGTH | 3744 | |
| | 50 | 0C | 50 | D1 | 004D3 | CMPL | LENGTH, 24(R10) | 3746 | |
| | 18 | AA | EA | 18 | 004D7 | BGEQ | 46\$ | | |
| | | | AA | DD | 004D9 | PUSHL | 12(R10) | 3764 | |
| | | 0C | AE | 9F | 004DC | PUSHAB | DSC | | |
| 00000000G | 00 | 1C | 02 | FB | 004DF | CALLS | #2, BASSRSET | | |
| | | | 009C | 31 | 004E6 | BRW | 51\$ | 3765 | |
| 00000200 | 8F | | 6B | D1 | 004E9 | CMPL | (R11), #512 | 3770 | |
| | | | 6D | 12 | 004F0 | BNEQ | 49\$ | | |
| | | | BA | 3C | 004F2 | MOVZWL | @12(R10), LENGTH | 3783 | |
| | 59 | 0C | 59 | D1 | 004F6 | CMPL | LENGTH, 24(R10) | 3785 | |
| | 18 | AA | 7B | 18 | 004FA | BGEQ | 50\$ | | |
| | | | AA | C2 | 004FC | SUBL2 | 24(R10), R9 | 3808 | |
| | 59 | 18 | 59 | D7 | 00500 | DECL | R9 | | |
| | | | 02 | C6 | 00502 | DIVL2 | #2, R9 | | |
| | 59 | 04 | AE | 18 | BA49 | MOVAB | @24(R10)[R9], T_LENGTH | | |
| | | | 10 | AE | B4 | 0050B | CLRW | T_DSC | 3809 |
| | 13 | AE | 02 | 90 | 0050E | MOVB | #2, T_DSC+3 | 3810 | |
| | | | 14 | AE | D4 | 00512 | CLRL | T_DSC+4 | 3811 |

| | | | | | | | | | |
|-----------|----|-----------|----|-------|-------|-------|--------|-------------------|------|
| | | 10 | AE | 9F | 00515 | | PUSHAB | T_DSC | 3812 |
| | | 08 | AE | 9F | 00518 | | PUSHAB | T_LENGTH | 3813 |
| 00000000G | 00 | | 02 | FB | 0051B | | CALLS | #2, STR\$GET1_DX | 3814 |
| 13 | AE | | 01 | 90 | 00522 | | MOVB | #1, T_DSC+3 | 3815 |
| | | 0C | AA | DD | 00526 | | PUSHL | 12(R10) | 3816 |
| | | 14 | AE | 9F | 00529 | | PUSHAB | T_DSC | 3817 |
| 00000000G | 00 | | 02 | FB | 0052C | | CALLS | #2, BASSRSET | 3818 |
| | 51 | 10 | AE | 9E | 00533 | | MOVAB | T_DSC, R1 | 3819 |
| | 50 | 18 | AE | 9E | 00537 | | MOVAB | DSC, R0 | 3820 |
| | | 00000000G | 00 | 16 | 0053B | | JSB | STR\$COPY_DX_R8 | 3821 |
| | | 18 | AE | 9F | 00541 | | PUSHAB | DSC | 3822 |
| | | | 6A | DD | 00544 | | PUSHL | (R10) | 3823 |
| | | | 6A | DD | 00546 | | PUSHL | (R10) | 3824 |
| 00000000G | 00 | | 03 | FB | 00548 | | CALLS | #3, STR\$CONCAT | 3825 |
| 13 | AE | | 02 | 90 | 0054F | | MOVB | #2, T_DSC+3 | 3826 |
| | | 10 | AE | 9F | 00553 | | PUSHAB | T_DSC | 3827 |
| 00000000G | 00 | | 01 | FB | 00556 | | CALLS | #1, STR\$FREE1_DX | 3828 |
| | | | 43 | 11 | 0055D | | BRB | 54\$ | 3829 |
| 00000400 | 8F | | 6B | D1 | 0055F | 49\$: | CMPL | (R11), #1024 | 3830 |
| | | | 2D | 12 | 00566 | | BNEQ | 53\$ | 3831 |
| | 50 | 0C | BA | 3C | 00568 | | MOVZWL | @12(R10), LENGTH | 3832 |
| | 18 | AA | 50 | D1 | 0056C | | CMPL | LENGTH, 24(R10) | 3833 |
| | | | 05 | 19 | 00570 | | BLSS | 50\$ | 3834 |
| | | 0C | AA | DD | 00572 | | PUSHL | 12(R10) | 3835 |
| | | | 11 | 11 | 00575 | | BRB | 52\$ | 3836 |
| | 50 | 18 | AE | 9E | 00577 | 50\$: | MOVAB | DSC, R0 | 3837 |
| | 51 | 0C | AA | D0 | 0057B | | MOVL | 12(R10), R1 | 3838 |
| | | 00000000G | 00 | 16 | 0057F | | JSB | STR\$COPY_DX_R8 | 3839 |
| | | 18 | AE | 9F | 00585 | 51\$: | PUSHAB | DSC | 3840 |
| | | | 6A | DD | 00588 | 52\$: | PUSHL | (R10) | 3841 |
| | | | 6A | DD | 0058A | | PUSHL | (R10) | 3842 |
| 00000000G | 00 | | 03 | FB | 0058C | | CALLS | #3, STR\$CONCAT | 3843 |
| | | | 0D | 11 | 00593 | | BRB | 54\$ | 3844 |
| | | 00000000G | 8F | DD | 00595 | 53\$: | PUSHL | #BAS\$K PRIUSIFOR | 3845 |
| 00000000G | 00 | | 01 | FB | 0059B | | CALLS | #1, BAS\$\$STOP | 3846 |
| 1B | AE | | 02 | 90 | 005A2 | 54\$: | MOVB | #2, DSC+3 | 3847 |
| | | 18 | AE | 9F | 005A6 | 55\$: | PUSHAB | DSC | 3848 |
| 00000000G | 00 | | 01 | FB | 005A9 | | CALLS | #1, STR\$FREE1_DX | 3849 |
| | 50 | | 01 | D0 | 005B0 | | MOVL | #1, R0 | 3850 |
| | | | 04 | 005B3 | | | RET | 3851 | |

; Routine Size: 1460 bytes, Routine Base: _BAS\$CODE + 094B

BAS\$\$FOR_INT
2-013

BAS\$\$FORMAT_INT - Basic format interpreter
End of module

J 11
16-Sep-1984 00:29:06
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFORINT.B32;1

Page 85
(12)

: 3190 3885 1 %SBTTL 'End of module'
: 3191 3886 1 END
: 3192 3887 0 ELUDOM

! End of module BASFORINT.B32

PSECT SUMMARY

| Name | Bytes | Attributes |
|------------|----------------------------------|-----------------------------|
| _BAS\$DATA | 24 NOVEC, WRT, RD ,NOEXE,NOSHR, | LCL, REL, CON, PIC,ALIGN(2) |
| _BAS\$CODE | 3839 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 | 19 | 0 | 581 | 00:01.1 |

COMMAND QUALIFIERS

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

: Size: 3578 code + 285 data bytes
: Run Time: 01:18.4
: Elapsed Time: 03:07.8
: Lines/CPU Min: 2974
: Lexemes/CPU-Min: 20864
: Memory Used: 491 pages
: Compilation Complete

