Laser-Scan Ltd.

LSLLIB

Technical Reference Manual

Issue 1.13 - 29-September-1993

The Facility Number Manager is: Paul Hardy
Consult him before allocating a new facility number for message definition.

Copyright (C) 1993 Laser-Scan Ltd
Science Park, Milton Road, Cambridge, England CB4 4FY tel: (01223) 420414

Document "LSLLIB REF", Category "Technical"
Document Issue 1.13 J Barber, TJ Ibbs, CC Brunt, RW Russell (modified 29-Sep-1993)

CHAPTER 7

USING /EXPC/ - TEXT EXPANSION

7.1 *Introduction*

These routines will carry out formatted expansion of binary data into character form. This is usually in order to produce an output line, and is often done by calling a routine (such as WRITEF or WRITAP) which uses them.

7.2 *Encoding into the output buffer*

len = EXPAND (format, arg1, arg2, arg3, arg4, ...)

out	- word	**len**	the length of the expanded string
in	- char	**format**	the format describing the required output
in	- variable	**arg<n>**	the arguments to fill out the format

The string format (usually a literal string) is copied into /EXPC/ character by character. If the escape character **%** is found, then the action determined by the sequence **%<escape sequence>** is obeyed - this may use an argument from the argument list following **format**.

Calling EXPAND with no arguments sets up a null line in /EXPC/ - a line of zero length. Calls to EXPAND reset the default state of the **%^** mode switches.

NOTE that EXPAND will check whether LSL_INIT has been called - see the description in Chapter 2 for details.

call APPEND(format, arg1, arg2, ...)

APPEND is the same as EXPAND, but appends the text to the end of the string already in /EXPC/.

Calling APPEND with no arguments produces undefined results. Calls to APPEND continue to use the values of the **%^** mode switches set up in earlier calls of EXPAND/APPEND.

7.3 *How the string in /EXPC/ is ended*

/EXPC/ is fully described in Chapter 6. As mentioned there, the actual text buffer, EXPBUF, is declared as one byte longer than would be expected from the

maximum length of line. This allows EXPAND/APPEND to output a final null byte at the end of any string.

Thus the string placed into /EXPC/ will always terminate in a zero byte, which will not be included in the count of characters held in EXPLEN.

Note that this zero terminating byte is not output when output is redirected to another destination (with %W or %WS - see below).

7.4 *EXPAND escape sequences*

The escape sequences allowed in the format string to EXPAND/APPEND are described below. Note that (unless otherwise stated) alphabetic escape sequences must be upper-case.

7.4.1 *Expanding a text string*

The following expand a byte buffer or string argument in place of the escape sequence.

%A<n> Synonym of %AZ

%AC<n> Includes the given ASCII byte array, assuming that the 1st byte is a count of the number of characters. Maximum length expanded is <n>, default 255

%AZ<n> Includes the given ASCII byte array, assuming that it is terminated by a zero byte. Maximum length expanded is <n>, default 255

%AD<n> Synonym of %S

%S<n> Includes the given ASCII string, assuming it is a character string. Maximum length expanded is <n>, default 255

%C Includes 1-4 ASCII characters (depending on current mode) from the argument - the characters are stored as bytes. Terminated by zero byte

%R Includes three radix-50 characters from the word argument

%5 Synonym of %R

%RZ,%5Z Effect as for %R or %5, but trailing spaces are suppressed

7.4.2 *Expanding integers*

The following expand an integer argument in place of the escape sequence.

If the output is signed (the default) then negative numbers are preceded by a minus sign. If the output is unsigned (set by the %U flag) then the number is output as an unsigned number - ie it is always positive. Positive numbers are never preceded by a plus sign.

A minus sign is included in the count of characters for the field width, and is always output adjacent to and immediately preceding the first digit output.

%I<n> Includes a decimal integer, field width <n> (default 6). If the integer won't fit in the field width, then it is output in the minimum field width that it will fit in.

%N<n> As %I, except that the default field width is 0

%O<n> Includes an octal number (default field 0). This is always assumed unsigned.

%X<n> Includes a hexadecimal number (default field 0). This is always assumed unsigned.

7.4.3 *Expanding real numbers*

The following expand a real (floating point) number in place of the escape sequence.

%F<f>.<d> Includes a floating decimal, where <f> is the total field width to use (including the decimal point), and <d> is the number of digits to be output after the decimal point. The default field values are 9.3

Note that there will always be at least one digit output before the decimal point, although there need not be any digits output after the decimal point. The sequence %F0.<n> can be useful for outputting in the minimum field width - compare with %I0 for integers.

Provided that the field (f) is large enough to contain the number, the number will be right justified in the field, and trailing zeroes will not be truncated. This means that the decimal points of numbers will always appear in the same character position.

If the field is insufficient to contain the number, then the number will be left justified, and will take up as many characters as required. Trailing zeroes may be truncated or not, depending on the use of %^T or %^P (default).

Thus the number 1234.5678 can be output as follows

```
with   %F        1234.568
with   %F0.3    1234.568
with   %F7.0      1235.
with   %F0.0    1235.
```

%E<d> Includes a real number with decimal exponent, where <d> is the number of significant digits to output - that is the number of digits between the decimal point and the 'E' delimiting the exponent. The default

field value is 4.

Thus the number 1234.5678 can be output as follows

```
with   %E      .1235E 004
with   %E0     .E 004
with   %E1     .1E 004
with   %E2     .12E 004
with   %E8     .12345678E 004
```

and for the number -1234.5678

```
with   %E      -.1235E 004
```

%G<f>.<d> Includes a general floating decimal. <d> significant figures are always output. Zero is output using %F<f>.<d>, numbers with absolute value less than 0.1 or greater than 10**<d> are output using %E<d>, while numbers within this range are output using %F<f>.<x>, where <x> (between 0 and <d>) is chosen to give <d> significant figures. The default field values are 9.3

%G is useful for outputting numbers where significance must not be lost, but greater readability than that provided by %E is desired. The sequence %G0.<n> can be useful for outputting in the minimum field width.

Thus the number 1234.5678 can be output as follows

```
with   %G        .123E 005
with   %G0.4     1235.
with   %G7.5      1234.6
```

7.4.4 *Expanding dates and times*

The following expand an (integer) argument representing a date or time in place of the escape sequence. There are routines to convert from standard VMS date/time strings to these date and time representations (see chapter 20).

A date is either the number of days since 17-NOV-1858, or if it is negative then it is a "delta date" i.e. the number of days from today.

A time is the number of 10 millisecond units since midnight.

%DD Translates the argument to a date, and includes it in standard VMS date format, e.g. either "22-OCT-1987" (for standard dates) or "54" (for delta dates)

%DT Translates the argument to a time since midnight, and includes it in standard VMS time format, e.g. "14:04:23.56"

7.4.5 *Expanding into a different destination*

The following redirect text expansion into a different buffer or string, instead of the default /EXPC/

%W Further output will be expanded into a user-specified buffer. The argument is the address of the required byte array.

Note that no overflow checking is performed - thus EXPAND/APPEND will quite happily attempt to write off the end of the buffer. This is accomplished by assuming that the destination buffer is of the same maximum length as EXPBUF - ie 1024 characters.

%WS As %W but the argument is the address of a string descriptor. Overflow checking is enabled - the expansion will stop when the string is full, and the string is padded with spaces (if necessary)

7.4.6 *Repetition*

The following cause a piece of text to be repeated in the expansion.

The enclosing repetition brackets must be matched within any call of EXPAND or APPEND (that is, %(with %) and %[with %]). There may be up to 64 repetition sequences (!) in any one call of EXPAND/APPEND, and these may be nested. If the number of repetition sequences is exceeded, then the string %<*> will be output instead of the offending repetition sequence.

%[...%] The enclosed section ("...") will be repeated. The argument is a word specifying how many times the section should occur in the expanded result. If the argument is less than 1, then 1 is assumed.

%(<n>...%) The enclosed section ("...") is expanded <n> times (default number of times is 1)

7.4.7 *Formatting*

The following influence how further escape sequence arguments will be output or interpreted

%P<c> Sets the padding char to <c> (default space). This character will be used in padding fixed field integers from now on. Note that it does not affect the placement of the '-' sign for negative numbers - these are still output directly preceding the integer.

%U Set unsigned mode for next integer - the next integer (only) output with %I or %N is output as an unsigned number.

%^B Set Byte mode - integer arguments are assumed to be bytes (BYTE or LOGICAL*1) from now on

%^W Set Word mode - integer arguments are assumed to be words (INTEGER*2) from now on

%^L Set Long mode (default) - integer arguments are assumed to be longwords (INTEGER or INTEGER*4) from now on

%^F Set single precision floating point mode (default) - real arguments are assumed to be single precision reals (REAL or REAL*4) from now on

%^D Set double precision floating point mode - real arguments are assumed to be double precision reals (REAL*8) from now on

%^P Set padding of numbers output using %F - numbers will always have the requested number of decimal places, including trailing zeroes.

%^T Set truncation of numbers output using %F (default) - numbers for which the specified field is insufficient will have any trailing zeroes after the decimal point truncated. The first digit after the point will never be truncated.

7.4.8 *Miscellaneous*

The following escape sequences do not take an argument

%M Multiplicity - if the last integer output with %I, %N, %O, or %X was not 1, then an 'S' is expanded, otherwise this sequence is ignored. The multiplicity is singular immediately after EXPAND or APPEND has been called - that is the default situation is as if 1 had been expanded.

%m As %M, but produces 's' (lower case letter)

%T Expands a tab character

%% Expands a single '%' character

% %<space> is ignored and may be used as a terminator

7.4.9 *Unrecognised escape sequences*

The following are used if an escape sequence is not recognised

%<unexpected char> is expanded as ?<unexpected char>

%^<unexpected char> is expanded as %^<unexpected char>

7.5 *Output routines using EXPAND/APPEND*

call WRITEF(<args as for EXPAND>)

This routine calls EXPAND on its arguments, and then calls TTWSTR. This latter call writes the contents of /EXPC/ to SYS$OUTPUT using VIO$PUT_OUTPUT.

call WRITAP(<args as for APPEND>)

This routine calls APPEND on its arguments, and then calls TTWSTR. This latter call writes the contents of /EXPC/ to SYS$OUTPUT, using VIO$PUT_OUTPUT.

The alias **WRTAPP** is provided for WRITAP.

7.6 *Routine to output angles*

char = DISPANG(secs, flg)

out - character*14	**char**	the angle in DD MM SS.SSS format. The format depends on the value of **flg**
in - real*8	**secs**	the angle to output in second
in - integer	**secs**	output format to use; see below

This routine is used to output an angle in the format that the routine READANG can read. See the chapter on reading numbers for details of these formats.

The value of **flg** should be one of the following values, as defined in LSL$CMNLSL:READANG.PAR

READANG_ANGLE	for angle with no hemisphere
READANG_LONGITUDE	for angle represents a longitude
READANG_LATITUDE	for angle represents a latitude