%WRAP MACRO TO OUTPUT TEXT DURING REPORT WRITING

Robert Musterer
Miles, Inc.

ABSTRACT:

Often a desired report layout does not allow for sufficient space to print out a long text string. The %WRAP macro was written to accommodate this. This macro wraps text within the confines of a defined width. Whenever possible, a clean break point is used to go to the next line, and all wrapped lines are indented two spaces. Three required parameters are: variable name, starting output column, and output width. Remaining parameters are all optional and include: name of variable for current line setting, additional variables to concatenate with required variable, formats for all variables, break point delimiters, variable name for end of file flag, and justification (overrides wrapping - cannot justify and wrap same text). Return of pointer control is handled by two variables, one indicates the starting or return line and the other indicates the last or next line.

INTRODUCTION:

Over time we are finding that we are entering and reporting more and more supportive text fields. SAS allows up to 200 characters in a single variable, but the print width we most commonly use is restricted to 132. Since the text is usually output alongside the other fields it supports, the available number of columns is greatly reduced. To facilitate printing full text in narrow output window, the macro %WRAP was written.

%WRAP is used during report writing (Data _NULL_; File PRINT;) and handles from one to eleven variables. There are only three required parms: the name of the variable containing the text to be wrapped, the starting column for positioning the output, and the display width to be used for the variable. The end result is that the text is output within the confines of the defined output columns, with wrapped lines indented two spaces, and words are not split across two lines unless the output width is shorter than the word itself. Pointer control is returned via two variables, RET_LN which indicates the initial starting line number, and NXT_LN which indicates the next line after the wrapped text. As %WRAP was used numerous enhancements have been added. The most important one is probably an option to control justification of the text. This option is considered to be mutually exclusive to wrapping, so the text may either be wrapped or justified - not both. The only data step requirements outside of the macro's control are that the FILE statement must use options N=PS and LINE=<ln> (If use a variable name other than LN, this must be indicated in the macro call).

THE MACRO ITSELF:

The entire macro consists of about 175 statements excluding all comments, and requires 6 pages to print (about 2 pages worth of comments). Space constraints therefore prohibit including the actual code and also a too detailed walk through. However the following summary of the major points should suffice.

There are 3 required positional parms and 26 optional key word parms.

Positional parms:
VAR name of primary variable being output
W_COL starting column for output, may be a constant or a variable name
W_WIDTH width of output window, may be a constant or a variable name

Key word parms:
LINE variable name for current line number, default is LN
FMT optional format for primary variable
XVAR1-XVAR10 additional variables to be concatenated as part of the total text string
XFMT1-XFMT10 optional formats corresponding to the additional variables
DELIM delimiters for determining wrapping break points.
JUST justification, options are L, R, C, and N
None is the default.
END variable name in END= SET statement option, default is EOF.
LN_CNT option to have macro determine number
of lines required rather than actually write out
text. Options are Y or N, default is N.

First the macro checks a status flag to see if this
is the first time the macro has been called for the
given data set. If yes, several temporary working
variables are created and assigned their
appropriate attributes. Next it checks if output
formats have been specified for the variables to
be output. If yes, the put function is used to put
the formatted output into a series of temporary
variables. If not, the values are left justified
and any trailing blanks are removed, and the result
is stored in the temporary variables. The lengths of
the resultant temporary variables are then
determined and stored in an array, and the total
length is calculated. If the total length is greater
than 200, then the text will be processed in 200
character increments. A nested macro
WWINDOW is used to combine the text from
multiple input variables into 200 character
parcels.

WRAP does not check to see if the wrapping
spans a page break. This is considered the job of
the overall data step. If WRAP is called with
optional parm LN_CNT=Y, then it only counts the
number of lines required so the data step logic
can act on this information to control page
breaks. This is appropriate since the macro has
no way of knowing if additional variables are also
being
wrapped by another call to %WRAP. Another
optional parameter controls justification. If the
text is justified it is not wrapped.

The next section of the macro is only executed if
the LN_CNT option is not chosen. This section
handles the justification. First the output width is
compared against the text length to assure it is
compatible, if not a warning message is written to
the log and a series of pound symbols (#) are
written in the output. If it is acceptable, the text
is output using the specified justification.

Otherwise the text is being output using the
wrapping algorithm. The 200 character parcels,
which result from macro WWINDOW referred to
above, are processed by cutting off appropriate
sized chunks. Each chunk is examined to test if
there was a clean break between two words. If
not, the pointer is backed up until a clean break
point is found and the smaller chunk is written
out. If the defined output width is too small to
accommodate a clean break point, then a word
will be split to force the wrap. This process is
cycled through until the entire text has been
output.

The testing for a clean break point is based on
looking for a blank or one of several special
characters. The delimiters used may be adjusted
by the use of an optional parameter named
DELIM. By default the delimiters are a blank, a
comma, a dash, a slash, an exclamation mark,
and an ampersand.

A very short sample program illustrates the use
of the WRAP macro.
DATA _NULL_;
FILE PRINT N=PS LINE=CURLN;
SET MYDATA END=THE_END;
PUT @1 MYVAR1 6.
@16 MYVAR2 5.2
@30 MYVAR3 5.3 @ ;
%WRAP(MYTEXT,45,30,L1NE=CURLN,END=THE_ END);
PUT #RET_LN @85 MYVAR4 ;
PUT #NXT_LN ;
RUN;