INTRODUCTION

A utility macro, REPLACE, is presented which can help avoid repetitious macro code. It will be discussed in the context of one of the programs which motivated its development. As a by-product of discussing the macro, various macro functions will be reviewed. In particular, the quoting functions %NRSTR and %UNQUOTE are demonstrated in a context which makes them a natural part of the macro facility.

The macro REPLACE is used to write a sequence of lines which are alike except for the occurrence of one sequence of characters. An important feature is that macro calls and macro variables are resolved after replacement when %NRSTR is used in the call. For example the call:

```
%REPLACE
  ( LIST= FIRST SECOND THIRD,  
    TARGET= %NRSTR 
      ( IF 0 < XXXX < 10 THEN %XXXX; ) 
  )
```

will produce the code:

```
IF 0 < FIRST < 10 THEN %FIRST ;
IF 0 < SECOND < 10 THEN %SECOND ;
IF 0 < THIRD < 10 THEN %THIRD ;
```

FIRST ATTEMPT - AN INTERNAL SOLUTION

A first attempt might be to set up an array of macro variables V1 to V8 representing the changing part of the line. Remember the ith element in the array is referenced by &&V&I. Thus we obtain:

```
%LOCAL I V1 V2 V3 V4 V5 V6 V7 V8 
/* INITIALIZATION */
%LET V1 = BN 
%LET V2 = CITP 
%LET V3 = CMDT 
%LET V4 = DITP 
%LET V5 = NOCT 
%LET V6 = OMSG 
%LET V7 = RPCD 
%LET V8 = STMN 
/* MAKE ARRAY STATEMENTS */
%DO I = 1 %TO 8 ;
    ARRAY &&V&I (*) %ARRAYLST 
      ( PREFIX=&&V&I, FROM=8808, TO=8907 ) ;
%END ;
```

This is one form of repetition. Each array statement is like the others; only the array name and the prefix change. Ahah! It could be handled in a %DO - loop, i.e., it can be controlled by rewriting the offending code. (Note: %ARRAYLST generates a variable list in which each variable name consists of a prefix followed by a year/month in the given range. But what %ARRAYLST does or how it is coded is really irrelevant to the problem of interest: controlling repetition.)

It has the advantage of simplicity, but there is still an element of repetition in the initialization. Moreover, it would require modification to use in other contexts. Adding a little more complexity helps.

The macro function %SCAN can be used to generate the variable part of the above loop. But now the macro function %EVAL is needed to control the index of the loop. Now we have:

```
%SCAN ( ) %ARRAYLST (  
  PREFIX=&&V&I,  
  FROM=8808,  
  TO =8907 ) ;
```
/* ------ SCANNING LOOP ------ */

%LOCAL LIST WORD I ;

%LET LIST = BN CITP CMDT DITP NOCT OMSG RPCD STMN ;
%LET I = 1 ;
%LET WORD = %SCAN (&LIST, &I) ;

%DO /* MAKE ARRAYS */
%WHILE ( %LENGTH (&WORD) > 0 ) ;

ARRAY &WORD (*) %ARRAYLST
( PREFIX=&WORD, FROM=8808, TO=8907 ) ;
%LET I = %EVAL (&I + 1) ;
%LET WORD = %SCAN (&LIST, &I) ;
%END ; /* MAKE ARRAYS */

Has repetition been eliminated? Suppose the plan as suggested above were carried out for the program in FIGURE 1 at the end of this paper. There would be a lot of very similar %DO - %WHILE SCANNING loops - one for the %TIMESHT calls, another for the array statements, and another for the array processing. Moreover, FIGURE 1 is only part of a larger program with several data steps similar to the one shown.

Each SCANNING loop has a core difference, but most of the code would be precisely the same because it has the same function. For example: in the SCANNING loop above, the core code is:

ARRAY &WORD (*) %ARRAYLST
( PREFIX=&WORD, FROM=8808, TO=8907 ) ;

How can the array statement be lifted out of the SCANNING loop? Now it is time to ask - "Can a macro handle the repetitious %DO - %WHILE loops?"

SECOND ATTEMPT - MOVE TOWARD AN EXTERNAL SOLUTION

It is difficult to remove the core code from the SCANNING LOOP because &WORD serves two masters:
- &WORD, an intimate part of the SCANNING LOOP, gives the value to be substituted.
- &WORD also indicates the code part to be replaced.

These functions must be separated before we can lift out the core code. This can be done by creating a new level of variable, known to the macro but not a macro variable. We will use XXXX for the new variable. Replace &WORD by XXXX in the core giving:

ARRAY XXXX (*) %ARRAYLST
( PREFIX=XXXX, FROM=8808, TO=8907 ) ;

The macro function %INDEX can be used to locate the first occurrence of XXXX, and %SUBSTR can be used to break up the line and piece it back together. For the first word scanned from the list of replacement values, this leads to the loop shown below.

%LOCAL X WORD TARGET FRSTPART LASTPART;

%LET WORD = BN ;
%LET TARGET = %STR ( ARRAY XXXX (*) %ARRAYLST
( PREFIX=XXXX, FROM=8808, TO=8907 ) ;
)

%LET X = %INDEX (&TARGET, XXXX) ;
%DO /* DO REPLACEMENT */
%WHILE ( &X > 0 ) ;
%LET FRSTPART = %SUBSTR
( &TARGET, 1, &X - 1 )
%LET LASTPART =
%SUBSTR ( &TARGET, &X + 4 )

%LET TARGET =
&FRSTPART.&WORD.&LASTPART
%LET X = %INDEX (&TARGET, XXXX)
%END ; /* DO REPLACEMENT */

Note that the core statement has been assigned to the macro variable, TARGET. This means that the core statement can be a parameter eventually. It is no longer a prisoner of the SCANNING LOOP.

There are, however, two things wrong with the above code.

1) %ARRAYLST will be executed when TARGET is defined. Hence &TARGET is really

ARRAY XXXX (*) XXXX8808 XXXX8809 XXXX8810 XXXX8811 XXXX8812
XXXX8901 XXXX8902 XXXX8903 XXXX8904 XXXX8905 XXXX8906
XXXX8907 ;

2) There is no provision for XXXX occurring at the beginning or end of the target string.

The macro call to %ARRAYLST can be hidden by using the quoting function %NRSTR to hide both % and & signs. In this case we have:

ARRAY XXXX (*) %ARRAYLST
( PREFIX=XXXX, FROM=8808, TO=8907 ) ;
%LET TARGET = %NRSTR
  ( ARRAY XXXX (*) %ARRAYLST
    (PREFIX=XXXX, FROM=8808, TO=8907)
  );
%
%SUBSTR will reveal the macro call hidden by
%NRSTR whenever the macro call is part of the
returned substring. The macro function %QSUBSTR
is needed. It leaves the created substring quoted.

Provision for XXXX at the beginning or end can be
made by allowing FIRSTPART and LASTPART to
have either the NULL value or the result of applying
%QSUBSTR.

When these features are added, the code is:

/* --- REPLACEMENT LOOP ---- */
%LOCAL
  X WORD TARGET FIRSTPART LASTPART ;
/* * INITIALIZATION */
%LET WORD = BN ;
%LET TARGET = %NRSTR
  ( ARRAY XXXX (*) %ARRAYLST
    (PREFIX=XXXX, FROM=8808, TO=8907)
  );
%LET X = %INDEX ( &TARGET , XXXX );
%DO /* DO REPLACEMENT */
  %WHILE ( &X > 0 )
  ;
  %IF &X > 1 %THEN
    %LET FIRSTPART = %QSUBSTR
    ( &TARGET, 1 , &X-1 );
  %ELSE
    %LET FIRSTPART = ;
  %IF &X+4< %LENGTH ( &TARGET ) %THEN
    %LET LASTPART = %QSUBSTR
    ( &TARGET, &X+4 );
  %ELSE
    %LET LASTPART = ;
  %LET TARGET =
    &FIRSTPART.&WORD.&LASTPART ;
%LET X = %INDEX ( &TARGET , XXXX );
%END ; /* DO REPLACEMENT */

Finally the result of the loop should be sent to the input
stack for macro execution. Now the macro function
%UNQUOTE is needed to allow % and & signs to
trigger macro activities. The extra line of code, placed
after the replacement loop, is

%UNQUOTE ( &TARGET )

It only remains to specify what the macro will do and
then to piece together the scanning loop and the
replacement loop into coherent macro code.

THE MACRO SPECIFICATIONS FOR %REPLACE

Two parameters are required:

1) The source string of replacement values as
given by LIST in the SCANNING LOOP
above.

2) The target string where re-placement takes
place as shown by TARGET in the
REPLACEMENT LOOP above.

In addition, the choice of XXXX is clearly one that
should belong to the calling macro, not %REPLACE.
Hence a third parameter, VARIABLE, is added to
allow the calling macro to assign any combination of
letters in place of the default, VARIABLE = XXXX.

The following calls may now be used to replace the
three blocks of repetitious code in FIGURE 1 at the
end of this paper:

%LET LIST =
  BN CITP CMDT DITP NOCT OMSG RPCD STMN ;
%REPLACE
  ( LIST = &LIST ,
    TARGET = %NRSTR
    ( %TIMESHFT ( PREFIX=XXXX, FROM=8703,
      TO=8808, LAST=8802 )
    )
  )
%REPLACE
  ( LIST = &LIST ,
    TARGET = %NRSTR
    ( ARRAY XXXX ( * ) %ARRAYLST
      (PREFIX=XXXX, FROM=8808,
      TO=8907 )
    )
  )
%REPLACE
  ( LIST = &LIST ,
    TARGET = XXXX (I) XXXX (I+1)
  )

To complete the macro it only remains to combine the
SCANNING LOOP and the REPLACEMENT LOOP.
The complete macro REPLACE is shown in FIGURE
2.

27
**IMPROVEMENTS**

The parameters LIST and TARGET can be quite long. Nothing above restricted TARGET to being one line of code. In the interest of readability, it is better to add one more level of indirection. Let the parameters LIST and TARGET name macro variables whose values hold the list and target strings.

This idea can be combined with the previous work by adding another parameter CALL, with the default CALL = DIRECT. When indirection is used then assign CALL = INDIRECT. For example the array call would be:

```
%LET VARS =
  EN CITP CMDT DITP
  NOCT OMSG RPCD STMN ;

%LET TARG = %NRSTR
  ( ARRAY XXXX (*) %ARRAYLST
    (PREFIX=XXXX, FROM=8808, TO=8907) ;
  ) ;

%REPLACE
  ( LIST = VARS ,
    TARGET = TARG ,
    CALL = INDIRECT )
```

To implement this improvement merely add the following lines at the beginning of the macro.

```
%IF &CALL NE DIRECT %THEN
  %DO ;
    %LET LIST = &&&LIST ;
    %LET TARGET = &&&TARGET ;
  %END ;
```

This implementation does not save the cost of passing large parameters, because the added %LET statements have the same cost. But this implementation is simple to execute and makes it possible to use %REPLACE in cases where the form in FIGURE 2 would not work because the parameters are too big. It is left to the interested reader to write the code making full use of indirection.

**CONCLUSION**

Using the suggested improvement, the code shown in FIGURE 1 may be rewritten as shown in FIGURE 3. Note that there is no duplication in this version. The most likely changes to the program are all at top housed in %LET statements, hence maintenance work has been simplified.

**ACKNOWLEDGEMENTS**

This paper was written while the author worked for the Atlantic Research Corporation Professional Services Group. FIGURE 1 is a simplified version of a program originally written under contract to the Bureau of Labor Statistics.

Comments are invited. A copy of the visuals shown at the SUGI conference may be from the author.

H. Ian Whitlock
Westat Inc.
1650 Research Boulevard
Rockville, MD 20850-3129
Phone: (301) 251-4347

28
DATA EVENTLSTM ( DROP = I ) ;
SET EVENTLSTM ( RENAME = )
  %TIMESHFT ( PREFIX=BN,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=CITP,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=CMDT,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=DITP,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=NOCT,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=OMSG,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=RPCD,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
  %TIMESHFT ( PREFIX=STMN,  
               FROM=8703,  
               TO=8808,  
               LAST=8802)
)
)
)
)
)
)
)
)
)
;

ARRAY BN (*) %ARRAYLST  
  (PREFIX=BN, FROM=8808, TO=8907)
;
ARRAY CITP (*) %ARRAYLST  
  (PREFIX=CITP, FROM=8808, TO=8907)
;
ARRAY CMDT (*) %ARRAYLST  
  (PREFIX=CMDT, FROM=8808, TO=8907)
;
ARRAY DITP (*) %ARRAYLST  
  (PREFIX=DITP, FROM=8808, TO=8907)
;
ARRAY NOCT (*) %ARRAYLST  
  (PREFIX=NOCT, FROM=8808, TO=8907)
;
ARRAY OMSG (*) %ARRAYLST  
  (PREFIX=OMSG, FROM=8808, TO=8907)
;
ARRAY RPCD (*) %ARRAYLST  
  (PREFIX=RPCD, FROM=8808, TO=8907)
;
ARRAY STMN (*) %ARRAYLST  
  (PREFIX=STMN, FROM=8808, TO=8907)
;

DO /* NO PRICE SENT MOVE INFO */
  /* FROM NEXT MONTH */
  I = 1 TO DIM ( DITP ) BY 2
  ;
  IF DITP ( I ) = ' ' THEN DO ;
    /* FILL IN GAPS */
      BN ( I ) = BN ( I+1 ) ;
      CITP ( I ) = CITP ( I+1 ) ;
      CMDT ( I ) = CMDT ( I+1 ) ;
      DITP ( I ) = DITP ( I+1 ) ;
      NOCT ( I ) = NOCT ( I+1 ) ;
      OMSG ( I ) = OMSG ( I+1 ) ;
      RPCD ( I ) = RPCD ( I+1 ) ;
      STMN ( I ) = STMN ( I+1 ) ;
  END ;
  /* FILL IN GAPS */
END;
/* NO PRICE SENT MOVE INFO */
/* FROM NEXT MONTH */

RUN ;
%MACRO REPLACE
( LIST = ,
  TARGET = ,
  VARIABLE = XXXX
)

%*----------------------------*;
%* PURPOSE: *
%* REPLACE ALL OCCURRENCES OF *
%* &VARIABLE IN &TARGET WITH *
%* EACH WORD IN &LIST. *
%* *
%* USE %NRSTR() AROUND TARGET *
%* STRING TO HIDE MACRO CALLS *
%* AND RESOLUTION OF MACRO *
%* VARIABLES UNTIL AFTER THE *
%* REPLACEMENT IS FINISHED *
%* *
%* EXAMPLE: *
%* %REPLACE *
%* (LIST= FIRST SECOND,
*  TARGET= %NRSTR(*
%* (IF 0 < XXXX < 10 THEN *
%*  %XXXX,)*
*  )*: *
%* PRODUCES *
%* IF 0 < FIRST < 10 THEN *
%* %FIRST ; *;
%* IF 0 < SECOND < 10 THEN *
%* %SECOND ; *;
*;
%* AUTHOR: IAN WHITLOCK, ARC *
%* DATE: JUNE 20, 1989 *
%*----------------------------*:
%LOCAL
WKTARGET WORD FRSTPART
LASTPART VARLEN X I ,

/* INITIALIZE */
/* EXECUTE LOOP */
/* VARIABLES */
%LET I = 1 ;
%LET WORD = %QSCAN ( &LIST , &I ) ;
%LET VARLEN = %LENGTH ( &VARIABLE ) ;

/* SCANNING LOOP */
%DO %WHILE ( %LENGTH ( &WORD ) > 0 ) ;

/* INITIALIZE INNER LOOP */
/* VARS */
%LET WKTARGET = &TARGET ;
%LET X = %INDEX ( &WKTARGET, &VARIABLE ) ;

/* REPLACEMENT LOOP */
%DO %WHILE ( &X > 0 ) ;
%IF &X - 1 > 0 %THEN
%LET FRSTPART = %QSUBSTR ( &WKTARGET ,
  1 ,
  %EVAL(&X-1) ) ;
%ELSE
%LET FRSTPART = ;
%LET X = %EVAL ( &X+&VARLEN ) ;
%IF &X <= %LENGTH(&WKTARGET)
%THEN
%LET LASTPART = %QSUBSTR ( &WKTARGET ,
  &X ) ;
%ELSE
%LET LASTPART= ;
%LET WKTARGET = &FRSTPART&WORD&LASTPART ;
%LET X = %INDEX ( &WKTARGET , &VARIABLE ) ;
%END ;

/* REPLACEMENT LOOP */
/*EXECUTE REPLACED CODE*/
%UNQUOTE ( &WKTARGET )

/* PREPARE FOR NEXT ITERATION */
/* OF REPLACEMENT */
/* LOOP */
%LET WKTARGET = &FRSTPART&WORD&LASTPART ;
%LET X = %INDEX ( &WKTARGET , &VARIABLE ) ;
%END ;/* SCANNING LOOP */
%MEND REPLACE ;
/* VARS FOR OLT */
%LET PREFLIST =
BN CITP CMDT DITP
NOCT OMSG RPCL STMN ;

/* RENAMES FOR EVENOTLT */
%LET ETIMSHFT = %NRSTR
( %TIMESHFT ( PREFIX = XXXX ,
                FROM=8703,TO=8808,LAST=8802) ;
)

/* ARRAYS FOR EVENOTLT */
%LET EARRAYS = %NRSTR
( ARRAY XXXX (*) %ARRAYLST
              (PREFIX=XXXX, FROM=8808, TO=8907) ;
)

/* FILL IN GAPS FOR NO PRICE */
%LET MOVEVALS = %STR
( XXXX ( I ) = XXXX ( I + 1 ) ;
)

DATA EVENOTLT ( DROP = i ) ;
  SET EVENOTLT ( RENAME = ( %REPLACE ( LIST = PREFLIST ,
                                 TARGET = ETIMSHFT )
                      ) ) ;
  %REPLACE (LIST=PREFLIST, TARGET=EARRAYS)
  DO I = 1 TO DIM ( DITP ) ;
  IF DITP ( I ) = '' THEN DO ;
    %REPLACE ( LIST=PREFLIST ,
               TARGET=MOVEVALS )
  END ;
  END ;
RUN ;