The Fast Track to Error-Checking Multiple Similar Screen Variables in a SAS/AF® Report Generator

Todd C. Folsom, Duke Power Company, Huntersville, NC

Introduction

Developers usually build SAS/AF® applications with distinct, dissimilar screen variables as needed for distinctly different kinds of information. The situation arose for me that I needed to use four to eight screen variables that were virtually identical except for their names. My application included two programs for users to build reports of the aquatic species found at various lake and river sampling locations. One program produces a report, or list, for one to four sampling locations in a portrait orientation. The other allows one to eight locations in a landscape orientation. The other allows one or more locations in a landscape orientation. The other allows one to eight locations in a landscape orientation.

The challenge here is that values of location come from the single variable Location and they will be ordered as columns (Fig. 1). Therefore the user must specify multiple values of location on the screen, and those screen variables must be validated with the one Location variable from the input data set. Figure 2 shows the SAS/AF® screen with variables Loc1, Loc2, Loc3, and Loc4. The user can enter values for any number (>=1) of these, in any order. All must be validated if modified, whenever the user presses Enter or End.

Figure 1. Observations from a SAS data set (left) are rearranged into the report format on the right, where the values from the variable Location become columns.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Location Species</th>
<th>Species</th>
<th>Red Lake</th>
<th>Blue Lake</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red crayfish</td>
<td>crayfish</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Red mayfly</td>
<td>mayfly</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Red caddis</td>
<td>caddis</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Red worm</td>
<td>worm</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>Blue mayfly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Blue worm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Screen for report generator showing multiple variables for location codes (Loc1-Loc4).

Step One- The INIT section

It would be conceptually easy, but tedious, to repeat the same block of validation code in the Screen Control Language (SCL) to call the DATALISTC selection box, if the screen value was not valid. This paper will show how to use arrays as a fast track to validating any number of similar screen variables using the same DATALISTC function.

Most of the input areas on either the portrait (Fig. 2) or landscape report generators are optional (only the number of locations and at least Loc1 are required). The user can choose whether to specify column headers and titles. Since blanks are valid here, and users may change their minds after entering values, special handling is necessary to assure that validation of the location codes does not interfere with the values for headers and titles. This will be explained, as well.

Figure 3. First section of code showing the INIT section. Note that line numbers were added here for easy reference from the text.

INIT:

* other stuff *
* Create non-window SCL vars for headers and make them "blank" for when a user does not put text in all header fields. Also create two or more blank titles; 1. control enter; 2. set up for error-checking the location codes; 3. dsid=Open('Work.Temp','r'); 4. localoc=varnum(dsid,'location'); /* find variable 'location' */ 5. array fld{*} S4 (loc1,loc2,loc3,loc4); 6. array loc{*} S5 loc1-loc4; 7. length h1-h8 S8 title1 title5 title6 $67; 8. RETURN;

SAS/AF programs, previously executed to call the species list report generators, created a temporary data set containing the sampling locations that the user wishes to work with. This data set, Work.Temp, is opened in INIT (Fig. 3, line 3), and it contains potentially numerous observations of species found at each of the several locations. (That is of no consequence to the programming, but it means the user will have to scroll through the DATALISTC selection box to find an occurrence of the desired location code. Usually there are single observations of the variable to be validated.) In line 4 of the program, the VARNUM function finds the variable Location in the data set for use later.

Two arrays are defined in the INIT section for the error-checking (Fig. 3, lines 5-6). Array Fld contains the names of the screen variables, and array Loc will contain the contents of the
Figure 4. The MAIN section that validates user input and passes non-blank screen values to other variables.

1. MAIN:
2. *validate locations entered by the user;
3. do i=1 to dim(Fld); /* names of screen variables for the locations */
4. if field('modified',Fld(i)) then
5. do;
6. rc=field('cursor',Fld(i));
7. /* check value of loc(i) in field Fld(i) against lookup file*/
8. if locate(dsid,loc,loc(i),'a')=0 then
9. do;
10. rc=field('erroron',Fld(i));
11. refresh;
12. loc(i)=datalist(dsid,1ocation,'Choose a valid location:');
13. rc=field('erroroff',Fld(i));
14. end;
15. end;
16. end;
17. end;

The beauty of this approach is that no matter how many of the location fields the user modifies, they will all be validated and with a minimum of execution time and coding effort. In addition, if you change the number of applicable fields on your screen, you only have to modify the array definitions in the INIT section.

The other component of the MAIN section deals with the header and title fields. Because these are optional on the report, it necessary to be sure valid values are passed to the SAS® code in the submit block. Problems will occur if, for example the SAS code says PUT @41 '&h11', and &h11 has no value. The solution is to define non-screen variables and use them in the SAS code.

The screen variables for the headers (Fig. 2), are numbered in matrix notation (row, column indicators). For example, h23 is the header for row 2, column 3. The screen variables are assigned to corresponding non-screen variables (h followed by a single digit) (Fig. 5, lines 2-9). The "If, Then, Else" condition in lines 2-9 states that, if the user did not put any text into the header field, set the non-screen variable to "" (missing). Otherwise assign the entered text to that non-screen variable. This works for the title variables, too (lines 11-13). The attribute of "" for an SCL character variable resolves in SAS code to "missing", and is therefore valid.

Todd C. Folsom
Environmental Division, (MGO3A3)
Duke Power Company
13339 Hagers Ferry Rd.
Huntersville, NC 28078 704-875-5426
tcfolsom@dpcmail.dukepower.com

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.