Programming Techniques Using Choice Groups with Extended Tables in Version 6 SAS/AF® Software

Richard A. Ragland, SAS Institute Inc., Cary, NC

Abstract

The use of choice groups with extended tables is a new feature of Version 6.06 SAS/AF software. A choice group is a list of mutually exclusive items that users may select. Extended tables are dynamic tables that display information. This paper will discuss programming techniques for generating dynamic selection lists to combine the features of choice groups with the features of extended tables. The discussion will include a more extensive definition of choice groups and extended tables, an explanation of how to set up choice groups and extended tables in a SAS/AF PROGRAM entry, and examples of Screen Control Language (SCL) programs that are easy to maintain and make applications more user-friendly.

Definition of Choice Groups

A choice group is a new feature of Version 6.06 SAS/AF software, and is an attribute of SASAF PROGRAM entries. A choice group is a set of fields that are similar in nature, and are referenced in SCL by a single choice group name. The choice group name, any valid SAS name, is defined in the Chole Group field of the attribute (ATTR) window of the PROGRAM entry for each choice group field in the display. Screen 1 shows a portion of a DISPLAY window showing a simple three field choice group. Screen 2 shows a portion of the ATTR window for a variable in the simple choice group. The ATTR window for variables &CHOICE2 and &CHOICE3 would be the same.

Linked Action Fields

Sometimes, the applications developer can make the display readable by associating an action field with each field in the choice group. The action field and associated field are linked action fields (LAF), and should be thought of as a single station within a choice group. The action field in a LAF can be a radio button or check box that is automatically filled in when the choice group station has been activated and the choice group given a value. A radio is defined by specifying a choice group name to multiple pairs of LAF's in the same choice group, and a check box is defined by specifying a unique choice group name to each LAF. Radios should be used for mutually exclusive selections, and check boxes should be used when more than one item can be selected.

To select, or activate, a radio button or check box, the user simply positions the cursor on either field in the LAF, and presses the enter key or mouse button. A check, 'X', or some other mark is placed in the action field, marking the associated choice group field as activated. Screen 3 is a portion of a DISPLAY window showing a three station choice group for radio buttons, and three, one station choice groups in a check box list. Notice that there is no difference in the way radio buttons and check boxes are designed in the DISPLAY window. The distinction is made in the ATTR window when choice group names are assigned to the various screen fields. All LAF's, the action fields and the associated station fields, have the same choice group name for the radio button choice group, and a different choice group name is given to every LAF in the check box choice groups. Screen 4 shows the ATTR window for an action field in a radio LAF. Table 1 and Table 2 show how the choice group names for the LAF's are defined differently in the ATTR window for radios and check boxes.
Radio LAF & Choice Group Name
& &RADIO1 RADIO
& &RADIO2 RADIO
& &RADIO3 RADIO

Table 1: Radio LAF's and choice group names

Check Box LAF & Choice Group Name
& &CHECK1 BOX1
& &CHECK2 BOX2
& &CHECK3 BOX3

Table 2: Check Box LAF's and choice group names

SCL Functions used with Choice Groups:

There are special SCL functions that are used to manipulate choice groups:

- **ACTIVATE**: Activates a field in a choice group. This function can also be used to "gray out" a field in the choice group, preventing a user from selecting that particular station.

- **ISACTIVE**: Determines whether or not a particular station is active.

Simple Choice Group Example:

Screen 5 is a SOURCE window showing a SCL program that checks active stations for a simple choice group named CHOICE, a choice group that uses the radio button LAF named RADIO, and choice groups that use the check box LAF named BOX1, BOX2 and BOX3.

```
mainlACTIvE = lsactive('OllCE');
if"CGj1SG = ' (Choice {lr'oup station II ACTIVE II ' is selected)';
  ACTIVIIE = lsactive('RADIO');
  if lsactive('BOX1') then CBlJSG = ' 1';
  if lsactive('BOX2') then CBlJSG = ' 2';
  if lsactive('BOX3') then CBlJSG = ' 3';
  CBlJSG = CBlJSG II ')
  return;
```

Screen 5: SOURCE window for manipulating Choice Groups.

Screen 6 is a TESTAF window showing the active stations for simple choice groups, radio buttons, and check boxes.

**Definition of Extended Tables**

An extended table is a feature of Version 6 SAS/AF software, and is a general attribute of SAS/AF PROGRAM entries. Extended tables display information in non-scrollable and scrollable areas of the screen. The information in the non-scrollable area can be instructions to the user about the application and/or screen variables that help manipulate the scrollable area. You define an extended table by selecting the "EXTENDED TABLE" field in the General attribute (GATTR) window of the SAS/AF PROGRAM entry.

Screen 7 shows a portion of the DISPLAY window, and Screen 8 shows a portion of the GATTR window for an extended table.

**Simple Three Station Choice Group**

- **Radio station 3 is selected**

- **Check boxes selected: 2 3**

Screen 8: Portion of a GATTR window for an extended table.

The DISPLAY design must contain three carets, **^**, in columns one through three, in order to separate the non-scrollable area from the scrollable area. These characters will not be displayed at execution time. Design the scrollable area as if there is only one row of information, keeping in mind that one row of information can occupy as many lines of text as exist on the DISPLAY. The information in the scrollable area can be a fixed or dynamic amount of
information such as:
• SAS data set variables
• SAS data view variables
• SCL variables
• values generated by SCL functions
• SAS macro variables
• simple text representing a selection list

SCL functions, Variables and Labels Used with Extended Tables.

There are special SCL functions, variables and labels specific to extended tables that are used to manipulate information in the scrollable area:

- **SETROW** function used to set the number of rows in the extended table.
- **TOPROW** function used to move a specified row in the extended table to the top of the scrollable area.
- **CURTOP** function used to determine the number of the row that is currently at the top of the scrollable area.
- **ISSEL** select list function used to determine if a row is being selected or unselected, or in what order the particular row was selected.
- **NSELECT** select list function used to determine the number of rows that have been selected.
- **SELECT** select list function used to force the selection of a specified row.
- **SELECTED** select list function used to determine the row that corresponds to the specified number of the selection order.
- **UNSELECT** select list function used to inactivate a specified row.
- **ENDTABLE** function used to determine the last row of a dynamic extended table.
- **_CURROW_** variable used to store the row number currently being processed by GETROW or PUTROW.
- **GETROW** SCL code within this statement label is executed as many times as necessary to fill the scrollable area with information.
- **PUTROW** SCL code within this statement label is executed whenever a row in the extended table is modified. This label is not required extended tables, but is necessary when your application processes modified values in the list.

Extended Tables

Selection lists and dynamic tables are two new types of extended tables now available in Version 6.06 SAS/AF software. For an extended table to be a selection list, you must specify a maximum number of selections and a selection order in the SETROW function. The selection order can be used to create different types of selection lists:

- **Y** Specifies that selected rows are to be moved to the top of the selection list in the order in which they are selected.
- **N** Specifies that a selected row is not to be moved to the top of the selection list.
- **A** Specifies that the selection list window is closed when the user makes a selection. This selection order is valid only when the number of selections has been set to one.

The following SETROW parameters will create a ten row selection list with a maximum of five selections, and every row selected by a user gets pushed to the top of the list in the order in which they are selected.

**CALL SETROW (10, 5, 'Y');**

When an extended table is a selection list the ISSEL, NSELECT, SELECT, SELECTED, and UNSELECT functions can be used in the application.

SCL has several built-in functions that bring up various selection lists. The following list of SCL functions are the available SCL selection list functions and a brief description of the kind of information in the list. For more information on these SCL functions see the SAS Screen Control Language: Reference, Version 6 First Edition.

- **CATLIST** Entries in a SAS catalog
- **DATALISTC** Data set observations for character variables
- **DATALISTN** Data set observations for numeric variables
- **DEVLIST** Supported hardware devices
- **DIRLIST** Members of a SAS data library
- **FILELIST** Current file lists
- **COLORLIST** Colors valid for the device
- **LIBLIST** Current libraries
- **LISTC** Character values
- **LISTN** Numeric values
- **SHOWLIST** One to thirteen specified character items
- **VARLIST** Variables in a SAS data set

A dynamic table is an extended table in which the maximum number of rows is not specified, and the dynamic parameter is set to "Y" in the SETROW function. The zero value for the number of selections, and the "1" for the selection order are selection list parameters, and are needed only as place-holders when defining a dynamic table.

**CALL SETROW (0, 0, "1", "Y");**

Use the ENDTABLE function in the GETROW section of the SCL program to indicate that there are no more rows.

Simple Extended Table Example

Screen 9 is a SOURCE window showing the SCL functions used to manipulate a dynamic table.

```
GETROW:
  N = _CURROW;
  if fetchobs (EHPID, _CURROW) = 1 then call endtable ();
  return;
```

Screen 9: SOURCE window for a simple dynamic extended table.
Screen 10 is a TESTAF window showing part of the resulting list of names in an employee data base.

<table>
<thead>
<tr>
<th>Command</th>
<th>Message</th>
<th>Press END to exit.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOTE: At bottom.</td>
<td>Enter employee name or partial name:</td>
<td>Press END to exit.</td>
</tr>
<tr>
<td>This is an example of a simple dynamic table</td>
<td>Choose a maximum number of employees to display:</td>
<td>Press END to exit.</td>
</tr>
</tbody>
</table>

Screen 10: TESTAF window for a simple dynamic extended table.

**Statement Label Execution Sequence**

The execution sequence for extended table SCL statement labels is as follows:

1. When the application is first invoked, the code in the INIT label will execute. If there is a call to the SETROW function in the INIT label, the code in the GETROW label will execute as many times as specified by the SETROW parameters or until the display is filled with information.
2. If the user scrolls forward or backward, the code in the GETROW label will execute as many times as necessary until there is no more information to be displayed or until the display is filled.
3. If the user modifies any of the fields in the extended table, the code in the PUTROW label will execute, and finally the code in the GETROW label will execute to rewrite the rows of information.
4. If the user presses the ENTER key and CONTROL ENTER or CONTROL ALWAYS has been set, the code in the MAIN label will execute and then the code in the GETROW label will execute.
5. If the user presses the END key and CONTROL ENTER or CONTROL ALWAYS has been set, the code in the MAIN section will execute and then the code in the TERM label will execute.
6. If the user presses the END key and CONTROL ENTER or CONTROL ALWAYS has not been set, only the code in the TERM label will execute.
7. If the user presses the CANCEL key, only the code in the TERM label will execute.

When using extended tables in an application, the number of items in a list can be determined during the execution of an application. For this reason, extended tables are recommended for an application when the number of fields in the list is likely to change.

**Example: Choice Groups with Extended Tables**

To combine choice group features with extended tables, enter a choice group name in the ATTR window's Choice Group field for one or more of the screen fields in the scrollable area of the display. Screen 11 shows the DISPLAY window of an example that uses an extended table.

**Screen 11:** DISPLAY showing an extended table with a LAF in the scrollable area.

The fields in the display are defined as follows:

1. &MESSAGE: a field used by the application to display messages about the status of the current selection list. This leaves the SAS system message line open for system environment messages.
2. &SEARCH: a field that allows the user to enter an employee name or a partial spelling of the name. Giving partial names is useful when the spelling of the name is uncertain, or when the user needs to process multiple employees with similar names.
3. &MAX1 - &MAX3: fields in the same choice group, the choice group name is MAXIMUM in this example. This choice group allows the user to determine the maximum number of employee names to display in the selection list by positioning the cursor on the desired maximum and pressing the enter key or mouse button.
4. &and &NAME: a linked action field in the scrollable area associated with the same choice group name, EMPINFO in this example, in the ATTR window. Using a check box choice group makes the execution of the application more readable.
5. &N: a field used to number the employees in the selection list.

The &SEARCH screen field is the only text entry field in the display. Reducing the number of text entry fields in an application reduces the chance of the user making a text entry error. By using choice groups and extended tables, the application developer can design the display with point and click fields, thus reducing the amount of text entry required.

This example displays a selection list of a subset of employee names from an employee data base. The list depends on a user entered search string and a maximum number of employee names to display. After the list is displayed, the user can:

1. Select an employee by pressing the enter key, or mouse button, on the employee's name. The LETTER SCL function, which is a call to the SAS/FSP® FSLETTER procedure, is then used to send a letter with the selected employee's personal information.
2. If the desired name is not in the current list:
   a. Choose a higher maximum number of employees to display by rerouting the cursor over the desired maximum and pressing the enter key, or mouse button.
   b. Re-enter another search string to bring a more efficient list of employees into the selection list.
3. END out of the application.

The SCL code shown in the Appendix drives this example. The SCL statement labels, functions and variables used to manipulate the choice groups and extended table in this example are discussed in detail as the example is explained below.

The default maximum values in the MAXIMUM choice group are stored in a data set, and during the execution of lines 00008 - 00011
of the INIT label, these values are placed in the respective screen fields, &MAX1 - &MAX3. The ACTIVATE function call on line 00012 gives the MAXIMUM choice group the value of the first station, &MAX1. Screen 12 shows the display as the user invokes the application. The initial message prompts the user for a name or a partial name, the default maximum values are in place, and &MAX1 activated. The SETROW function call on line 00014 in the INIT label uses the new extended table features to set up a selection list. The code in the GETROW label executes, but since the number of rows is set to zero in the SETROW function, no rows are displayed.

Screen 12: Choice group with extended table example after the INIT label execution.

After the user enters an employee name or partial name, 'Jones' in this example, the code in the MAIN label executes. Lines 00023 - 00028 execute a SUBMIT block and create a subset data set of the employee database. The WHERE data set option on line 00025, a new data set option in Version 6.06 Base SAS® software, is used to subset a data set depending on the expressions in the WHERE clause. The WHERE clause in this particular example creates observations in a subset data set of the employee database, only if the value of the screen variable &SEARCH is contained, "?", anywhere within the data set variable NAME. Once the subset data set has been created, lines 00027 - 00050 in the MAIN label execute to determine how many rows to place in the selection list, and determines which message the user should see in &MESSAGE.

Basically, two questions are answered when lines 00027 - 00050 in the MAIN label executes:

1. Does a search string, or employee name, exist to process?
2. How does the number of observations in the subset data set relate to the current value of the MAXIMUM choice group?

Screen 13 shows the display after the user has entered a search string in the screen variable &SEARCH, using the default MAXIMUM value of one.

Screen 13: Choice group with extended table example after search string has been entered.

The number of rows in the list during the execution depends on a value selected from the MAXIMUM choice group in the non-scrollable portion of the display, or from the number of employee names that satisfy the user entered value in &SEARCH. If the user is unsatisfied with the employee names in the selection list, then choosing a new MAXIMUM may display the desired name. By positioning the cursor over a higher MAXIMUM and pressing enter, the user can add more employee names to the list. The lines 00033 - 00056 in the MAIN label will execute and a new value for the number of rows will be used in the SETROW function on line 00053.

Screen 14 shows the display after the user has modified the MAXIMUM choice group to increase the number of rows displayed in the selection list.

Screen 14: Choice group with extended table example with a new MAXIMUM number of rows.

When the employee's name has been located the user can then select that employee by positioning the cursor over the name and pressing the enter key. The LAF becomes activated causing the code in the PUTROW label, lines 00073 - 00078, to execute. On line 00073, the row that has been modified is checked to see if it has been selected, or activated. If the row has been activated, the choice group, EMPINFO in this example, must be given a value. Line 00074 gives EMPINFO the current value of the screen variable &NAME. Once the choice group has a value, the application can continue to send the message in the letter to the selected employee, lines 00075 - 00076. Screen 15 shows the example letter that has been created.

Screen 15: Choice group with extended table example document containing employee information.

After the letter is sent, the application returns to the customized display. The code in the GETROW label executes to re-display the screen, and the row containing the activated row is
marked as selected. Screen 16 shows the application's screen after sending the letter. At this point the user can repeat any of the above steps to send a letter to another employee, search for another employee name, or END out of the application.

Screen 16: Choice group with extended table example showing selected employee name.

SCL code could be added to the MAIN label to allow the user to modify the values in the MAXIMUM choice group fields, and save any modifications to the data set. Saving modifications to the MAXIMUM choice group allows the user to maintain his/her own environment for using this application. SCL code could also be added to the PUTROW label so that if an employee's name changes, the changed name could be saved to the data set and a letter would be sent containing the new changes. A SAS data set should never be opened in the GETROW label because of the multiple executions.

Summary
The combination of choice groups, extended tables and Screen Control Language allow the SASIAF applications developer great flexibility in designing applications that are user-friendly and easy to maintain. Through the use of examples, this paper discussed Screen Control Language programming techniques for generating dynamic selection lists by combining the features of choice groups with the features of extended tables.

Acknowledgements
The author wishes to thank Pat Berryman, Stuart Austin, Jama Lidral, Tom Sherron, Anna Kushner, Deva Kumar and Glen Walker for technical review, and Paul Kent for aid in typesetting the paper.

References


Base SAS®, SAS/AF®, and SAS/FSP® are registered trademarks of SAS Institute Inc., Cary NC, USA.

Appendix
SCL code for the Employee Information Database application:

```
!_SAS-BUILD-SOURCE-Tabla12PProgram(F)
Flld UnitViewLocalsGlobalsMain
!-!
0001:INIT OLIMAX = 30; OBSEARCH = "<INIT>"; EMPID = 0;
0002: MESS = "Please enter the employee name to search:";
0003: MSC1 = "An employee name found. Try again:";
0004: MESS3 = "No employee names exist. Choose a higher max.:
0005: MESS4 = "Search string is inefficient. Try again:";
0006: MESSAGE = "MESS;
0007: MOV1B = open ("SUGI.MOV1A", "R");
0008: call set (MOV1B);
0009: do i = 1 to 3: RC = fetchobs MOV1B, i; end;
0010: RC = close (MOV1B);
0011: RC = activate ("MAIN", 1);
0012: replace WHERE "USERID";
0013: call setrow (EMPINFO, 1, "name");
0014: return;
0015: call new (MESS1);
0016: end;
0017: end;
0018: end;
0019: if OBSEARCH = "SEARCH" & OBSEARCH = "" then do:
0020: OBSEARCH = "SEARCH";
0021: EMPID = 0;
0022: RC = delete ("SELECT");
0023: submit continue:
0024: data SELECT (keep = NAME);
0025: set SUGI.SELECT (where = (NAME = "JONES")); run:
0026: explore:
0027: EMPID = open ("SELECT", "R");
0028: call set (EMPID);
0029: do i = 1 to 3: RC = fetchobs EMPID, i; end;
0030: call set (EMPID);
0031: return;
0032: end;
0033: MESS1 = "NAME";
0034: if OBSEARCH = "" then:
0035: NAMELIST = 0;
0036: MESSAGE = "MESS1;"
0037: end;
0038: end;
0039: if not NAMELIST then do:
0040: RETURN;
0041: MESSAGE = "MESS1;"
0042: end;
0043: else if OBSEARCH in ("MAXIMUM", "MORE") then do:
0044: NAMELIST = 0;
0045: MESSAGE = "MESS1;"
0046: end;
0047: if NAMELIST = 1 then:
0048: if ACTIVE = 3 then MESSAGE = "MESS4;"
0049: end;
0050: end;
0051: if OBSEARCH = "SEARCH" & OBSEARCH = "" then do:
0052: call setrow (EMPINFO, 1, "name");
0053: NAMELIST = 1;
0054: OLIMAX = OLIMAX;
0055: end;
0056: return;
0057: end;
0058: end;
0059: if OBSEARCH = "SEARCH" & OBSEARCH = "" then do:
0060: RETURN;
0061: RC = delete ("SELECT");
0062: end;
0063: getrow: if EMPID then do:
0064: RC = fetchobs (EMPID, _currOBS_);
0065: _currOBS_ = 1;
0066: if NAMELIST then do:
0067: RC = activate ("EMPINFO", 0, _currOBS_);
0068: RC = unselect (_currOBS_);
0069: end;
0070: end;
0071: return;
0072: end;
0073: putrow: if OBSEARCH = "SEARCH" & OBSEARCH = "" then do:
0074: EMPID = NAME;
0075: DS = SUGI.SELECT (where = (NAME = "William"));
0076: call letter ("SUGI.SELECT.EMPLOYEE.LETTER", "print", DS);
0077: end;
0078: return;
```

1557