Using Extended Tables in Version 6 SAS/AF Software
Glenn H. Ilano, SAS Institute Inc., Cary, NC

Abstract

Extended tables are a feature of Version 6 SAS/AF software. This tutorial describes the requirements for their use, how to create a SAS catalog PROGRAM entry that can be used as an extended table, special features of screen control language (SCL) that support extended tables, and the execution sequence for an extended table program. Three sample extended table programs are presented and discussed.

Introduction

Extended tables are displayed by a special kind of Version 6 SAS/AF PROGRAM entry. The typical PROGRAM entry has a display of fixed text and variable fields. The extended table PROGRAM display has a scrollable area whose contents are generated by the underlying screen control language (SCL) program. The directory of a SAS catalog being displayed by PROC BUILD can be thought of as a functional extended table. It has some fixed text at the top of the display and a body of vertically scrollable information.

An extended table is an information display. It is not a form of data storage, such as a SAS data set or database table. However, because an extended table is a means of displaying information, that information can be

- in a SAS data set
- in an SCL variable array
- generated by SCL statements
- imported from macro variables.

In this tutorial, the following topics will be discussed:

- how to create an extended table PROGRAM entry
- special features of screen control language for extended tables
- SCL code execution sequence for extended tables
- three examples of extended tables.

Creating the Extended Table Display

Extended tables, as with all SAS/AF entries, must be created with PROC BUILD. To allow a PROGRAM entry to be used as an extended table, first edit a PROGRAM entry, then open the GATTR window and mark the special extended table attribute (Screen 1).

### Screen 1. The GATTR window.

In addition to specifying that a PROGRAM entry is an extended table, you must design the display portion of the entry. The display of an extended table consists of two distinct regions; a nonscrollable and a scrollable section. They are separated by three carets (h h h) in columns 1 through 3. Both sections can have fixed text and screen variables. Screen 2 depicts the display portion for a simple extended table. The scrollable area will consist of repeated values of screen variables NUM and SQUARE.

### Screen 2. The display window of a simple extended table.
Extended Table SCL

Once a PROGRAM entry has been defined as an extended table, appropriate SCL code must be written and compiled to support window processing. As with all SAS/AF PROGRAM entries, reserved statement labels are used to denote sections of SCL code that execute at specific times.

init: labels statements executed when the program is first invoked.

main: labels statements executed during normal execution of the program, typically when a user modifies a screen variable and presses ENTER.

term: labels statements that execute when the END and CANCEL commands execute.

Two additional reserved statement labels are available for extended table PROGRAM entries.

getrow: labels statements executed as many times as necessary to display a full window of information. GETROW is required for extended tables and is always the final reserved label executed when an extended table is displayed. Note that this means that any variable values updated in GETROW are not available to INIT or MAIN until the next execution cycle.

putrow: labels statements executed for rows modified by the user. It is required if the scrollable portion of the extended table has fields that are not protected and you have statements to be executed when the fields are modified. PUTROW executes before MAIN or TERM, so any variable values updated in PUTROW are available during that execution cycle.

Two SCL CALL functions are available for use with extended tables:

- **SETROW** sets the number of rows in the extended table. You must use the SETROW function if you wish to have a useful extended table. SETROW cannot be used in the GETROW and PUTROW sections.

- **TOPROW** causes the extended table to be scrolled to the specified row. TOPROW cannot be used in the GETROW and PUTROW sections.

Additionally an automatic variable is available for use exclusively with extended tables:

- _CURROW_ is set to the row number being processed by SETROW or PUTROW.

Execution Sequence

The general sequence of execution for the reserved SCL labels in an extended table is as follows:

When the program is first invoked,

- INIT is executed.

If the SETROW function is called and the number of rows is set to a number greater than 0,

- INIT is executed.
- GETROW will be executed at least once, for _CURROW_ equal to 1. If the SETROW function established an extended table of more than one row, GETROW is executed repeatedly, until either a full window of rows has been written out or the final row in the extended table has been written.

PUTROW is not executed during the initialization phase.

If the user presses a vertical scrolling key or enters a vertical scrolling command, such as DOWN or BOTTOM, GETROW will again be called upon repeatedly to write a full window of extended table information.

If the CONTROL statement has been used to set CONTROL ENTER or CONTROL ALWAYS and the user presses ENTER,

- MAIN is executed
- GETROW is called upon to rewrite the extended table rows.

If there are any unprotected variables and the user modifies one or more of them and presses ENTER, the above sequence of events occurs,
independent of the current status of the CONTROL statement.

Finally, if any variables in the scrollable area of the display are modified and the user presses ENTER,

- PUTROW is executed for each modified row
- MAIN is executed
- GETROW is called upon to rewrite the extended table rows.

When the user executes the END command and if the CONTROL statement has been used to set CONTROL ENTER or CONTROL ALWAYS, or if the user has modified any field variables,

- MAIN is executed
- TERM is executed.

If the user executes the END command and if any variables in the scrollable area of the display are modified,

- PUTROW is executed for each modified row
- MAIN is executed
- TERM is executed.

If the user executes the END command, the CONTROL statement has not been used to set CONTROL ENTER or CONTROL ALWAYS, and the user has not modified any screen variables, only TERM is executed.

If the user executes the CANCEL command, no matter what other conditions are in effect, only TERM is executed.

Example 1: Numbers and Their Squares

Screen 2 is the display for the first example. In this program, variables NUM and SQUARE have the following field attributes: numeric, protected, and right justified.

The SCL code for this example is displayed in Screen 3. This program defines an extended table of 50 rows. It will display the row number and the square of that number as the scrollable text of the extended table. Although this is not a very practical example, it allows us to explore some characteristics of the extended table.

```
BUILD: SOURCE EXAMPLE1.PROGRAM (E)--->
Command ==> 
00001 init: 
00002 call setrow( 50 );
00003 return;
00004 main: 
00005 return;
00007 term: 
00009 return;
00010 getrow: 
00012 num := CURROW ;
00013 squared := num ** 2;
00014 return;
00016 putrow: 
00017 return;
```

Screen 3. SCL code for an extended table.

When this program is first invoked, INIT executes and uses the SETROW function to establish an extended table of 50 rows. Then GETROW executes five times, creating the display depicted in Screen 4. The panel is displayed and awaits user input.

```
Extended table example:
Integers and their squares.

<table>
<thead>
<tr>
<th>Number</th>
<th>Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
</tbody>
</table>
```

Screen 4. The executing extended table program.

If you use the DOWN command to scroll forward, the text in the upper portion of the display area remains the same. GETROW is now executed for rows 6 through 10, resulting in new numbers and squared being displayed (Screen 5).

MAIN, PUTROW, and TERM do not perform any function in this program.

When the user executes the END or CANCEL command, the window simply closes. You do not have to perform any special cleanup to end an extended table program.
Example 2: Editing a SAS Data Set

This example shows how to edit a SAS data set using an extended table.

SAS data set SUGI.NAMES has an unknown number of observations and three variables, FNAME, LNAME, and PHONE.

The display window for this example is shown in Screen 6. Variables FNAME, LNAME, and PHONE are all used to build the extended table rows. All three variables have the following field characteristics: character and unprotected.

There is an unprotected action variable, FIELD1, denoted by the ampersand (&) next to the text ‘Add observation?’ in the nonscrollable portion of the window. The SCL program will use this variable to determine if the user wishes to add an observation to the data set.

The entire SCL program is shown in Screen 7.

Note: In this program, there are no statements that check the validity of return codes. This was done to reduce the number of SCL statements being discussed, but is not necessarily recommended for actual applications.

In INIT, SUGI.NAMES is opened in update mode, allowing the program to add observations and change variable values. The SET function is then invoked, linking SAS data set variables FNAME, LNAME, and PHONE with the SCL variables of the same name on the panel. The SETROW function then establishes the initial size of the extended table to be equal to the number of observations in SUGI.NAMES.
As was discussed earlier, after INIT has finished processing, GETROW is repeatedly executed to build the rows of the extended tables. In this example, GETROW simply retrieves the observation whose number corresponds to the row being requested in the CURROW variable. Because the SET function had been used earlier, the program does not have to use the GETVARC function to retrieve the variable values.

The first statement in MAIN checks to see if the END command is being processed. If so, it is exiting MAIN to go directly to TERM, rather than possibly adding an observation to the data set.

If the program detects that the user wishes to add an observation by typing an 'X' in FIELD1, the program executes:
- FIELD1 is reset to blank
- the field variables, FNAME, LNAME, and PHONE, are set to blank
- an observation of blank values is added to the data set
- the TOPROW function is called to assure that the first row being displayed will represent the first observation in the data set
- an indicator variable, UNSORTED, is set to 1.

If the UNSORTED variable has a value of 1, then the data set is sorted alphabetically. If a blank observation has been added to the data set, the sort will place the new observation as the first observation in the data set.

PUTROW executes if one of the extended table variables, FNAME, LNAME, or PHONE has been modified. The UPDATE function will write any changes to SUGI.NAMES; otherwise any modifications would be lost. If either FNAME or LNAME had been modified, the data set is assumed to be no longer in alphabetical order and UNSORTED is set to 1.

SAS data set, SUGI.NAMES, is closed in the TERM label.

Example 3: A Simple Selection List

SAS/AF software provides SCL functions that present selection lists. Some examples are the LIBLIST, DIRLIST, and VARLIST functions. In this example, an extended table will be used to create a cursor sensitive selection list that allows a user to select a color, perhaps for a SAS/GRAPH title. The user will simply have to place the cursor on a color name and press ENTER to make a selection.

The display window for this example is shown in Screen 8. Variable ROWTEXT is an unprotected character variable.

```
SCREEN 8. The display for the extended table selection list example.
```

The entire SCL program is shown in Screen 9.

```
BUILD: DISPLAY EXAMPLE3.PROGRAM (E)
Command ===>
Please make a selection.

ROWTEXT

SCREEN 9. SCL code for the extended table selection list example.
```
The choices for colors are in an SCL character array, COLOR. Note that the array sets the initial values for the colors. CHOICE, a character variable that will contain the current selection, is initialized with a LENGTH statement.

In INIT, CONTROL ENTER is set. This allows the program to execute MAIN whether or not any fields are modified. SETROW is called to create an extended table that has as many rows as there are elements in the COLOR array. Note that if you choose to add more colors to the array, you will not have to change any SCL code other than the ARRAY statement.

GETROW simply reads the elements of the array into the rows of the table. The executing program is depicted in Screen 10.

Screen 10. Executing extended table selection list example.

If table row is modified, this program assumes that the user wishes to select that row. Since PUTROW executes for all modified rows before MAIN, variable CHOICE is set to the selected color if a field has been modified. In MAIN,

- the program first checks to see if CHOICE has a value. If it does, the program branches to TERM.
- the CURWORD function is used to save the text string under the cursor when ENTER was pressed
- the IN operator is used to determine if the word in variable CHOICE matches any of the elements of array COLOR. If it does, the program branches to TERM.
- if the program never branches to TERM, variable CHOICE is reset to blank.

In TERM, the macro variable CHOICE is set to the current value of the SCL variable CHOICE. The program is then unconditionally terminated by setting variable _STATUS_ to 'H'.

This program is designed to be called from another SCL program. Sample SCL code for using this example to select a color is shown in Screen 11.

Screen 11. Calling the extended table selection list.

Summary

Extended tables allow the SCL programmer to extend the capabilities of Version 6 SAS/AF applications. They are a means of displaying information, not of storing it.

Five elements of SCL are devoted exclusively to extended tables: statement labels GETROW and PUTROW, the SETROW and TOPROW functions, and the CURROW automatic variable.

PUTROW executes when a table row has been modified by user input, in which case it is the first reserved label to execute. This means that any information obtained in PUTROW is available for MAIN or TERM.

GETROW executes repeatedly to write the rows of the extended table. It is the last label to execute before the window is displayed. This means that any information obtained in GETROW is not available for INIT, MAIN, or TERM.

A developer can write extended table programs to perform as a customized cursor sensitive selection list, edit multiple observations in a SAS data set on one window, or simply display observations from one or more SAS data sets as rows in an extended table.

SAS, SAS/AF, and SAS/GRAPH are registered trademarks of SAS Institute Inc., Cary NC, USA.