Windows without Pains

Ross Z. Merlin

"Windows without Pains" was presented at SUGI 13 as a PC Hands-on Workshop. The following notes and examples were used in the presentation.

The windows that we will be learning about are not the three DMS (Display Manager System) windows that you see when you start SAS [1]; we will be learning about windows that are controlled by the DATA step. You decide what these windows look like, where and when they appear, what text and variables are displayed, and which variables will accept input from the window or be for display only.

To get an idea of what can be done with windows, execute the SUGIHILO program that is in the SAS sample library. On a typical PC, the file would be C:\SAS\SASSAMPL\SUGIHILO.SAS (not all installations choose to install the sample library; check with your installation representative). If the file date is 1-15-88, you should first make the following changes to the program: in the line that begins "window cheat", change rows=1 to rows=3; "high" 7/8; "low", 6/8; "end" 7/8; and "outof" 8/9. After you play the game a few times, try "27511" as the first guess (perhaps the magic number will change to "27512" in future versions).

To use windows, we only have to learn two statements: WINDOW, which defines a window; and DISPLAY, which displays a window and accepts input (data or commands).

```
data _null_;        * hello1.sas ;
   window hello
     #1 @30 'Welcome to  SUGI 13'
     #3 @30 '  Orlando Florida  '
     #5 @30 'March 27 - 30, 1988';
   display hello;
   stop;
 run;
```

Figure 1

The program in Figure 1 defines a window named "hello", displays it, then stops. The name for the window follows the same rules as for SAS variable names. The "#" and the "@" specifications give the line and column locations at which the following text or variable is to be displayed. The "#" or "@" can be followed by a positive integer (as in the example); a numeric variable, such as #listvar); or an expression that evaluates to a positive integer, such as @(margin+5). If the expression form is used, enclose the expression in parenthesis. Other pointer controls that can be used are "/", to advance to column 1 of the next line; and "+n" (where n may be a positive integer or numeric variable), to skip n columns.

---

1. SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.

The DISPLAY statement causes the text to be displayed; execution pauses until you press enter. The STOP statement is needed because the DATA step normally continues looping until an end-of-file or end-of-dataset condition occurs; since this step does not read a file or dataset, the STOP statement is needed to keep the program from looping forever. If you find yourself stuck in an infinite loop, you may be able to break out by entering the "END" command on the command line at the top of each window.

The program in Figure 2 adds attributes to the displayed fields. These and similar specifications are referred to as field options, since they pertain to the field (variable name or text string) that they follow. An attribute is specified by "attr=" or "a=" followed by one or more of the following: highlight, blink, rev_video (reverse video), or underline. If more than one attribute is to apply to a field (as in the tenth line of Figure 2), join the attributes with a comma and enclose in parenthesis.

```
data _null_;        * hello2.sas ;
   window hello
     #1 @30 'Welcome to  ' attr=highlight
            'SUGI 13' a=blink
     #3 @30 '  Orlando Florida  '
            attr=rev_video
     #5 @30 'March 27 - 30, 1988'
            attr=underline
     #7 @30 'PC Hands-on Workshop'
            attr=(rev_video,blink)
     ;
   display hello;
   stop;
 run;
```

Figure 2

Not all attributes will show on all monitors; "highlight" and "underline" did not work on some color monitors, but worked properly on monochrome monitors.

The specification that follow "window hello" in Figure 3 are called "window options". They apply to the entire window, not the individual fields.

"color=" -- background color:white, green, red, cyan, blue, black, magenta, yellow, gray.

"rows=" -- the number of rows (lines) within the window (excluding the borders).

"columns=' -- the number of columns used by the window.

"irow=' -- the initial row of your window. This is the location of the command line, not your line #1 (except that irow=0, 1, or 2 results in irow=2 since the border is in row 1).

```
data _null_;        * hello3.sas ;
   window hello
     color=blue rows=9 columns=40
      irow=11 icolumn=21
```

```
    #1 @10 'Welcome to  ' attr=highlight
           'SUGI 13' a=blink  color=red
    #3 @10 '  Orlando Florida  '
           attr=rev_video  color=yellow
    #5 @10 'March 27 - 30, 1988'
           attr=underline  color=green
    #7 @10 'PC Hands-on Workshop'
           attr=(rev_video,blink)
    ;
  display hello;
  stop;
run;
```

Figure 3

Also shown in Figure 3 is the field attribute "color=", which specifies the color for the field that it follows. The choices are the same as for the "color=" window option.

Another window option, "keys=", lets you specify a filename for a dataset that contains your own function key definitions.

Four additional field options are PERSIST=, AUTOSKIP= (or AUTO=), REQUIRED=, and PROTECT=. Each of these can have values of "yes" or "no". Usually a DISPLAY statement clears the screen before each execution. Any field with PERSIST=YES will not be cleared (unless it is overlaid); it will continue to show on subsequent executions of the DISPLAY statement. This may be useful when the location of a field changes from one DISPLAY to the next. The persisting field will remain visible until the end of the DATA step, it is overlaid by another field, or it is cleared when the same window is displayed with the BLANK option (discussed below). AUTOSKIP=YES causes the cursor to jump to the next input field when a character is typed in the last position of the current field. If AUTOSKIP=NO (the default) is in effect, the cursor remains at the last position of the field until the ENTER key is pressed. REQUIRED=YES means that data must be entered in the field before the user can proceed to the next observation. PROTECT=YES allows a field to be displayed but not modified.

The DISPLAY statement names the window to be displayed. There are three options: NOINPUT, BLANK, and BELL. NOINPUT means that none of the fields are to be used for input. The user can not change any of the displayed values. The user does not have to press ENTER to proceed to the next window; execution continues with the statement following the DISPLAY. Blank clears the window before the new values are displayed; this may be used to nullify the effect of the PERSIST=YES field option described previously. BELL causes the PC to beep when the window is displayed; this can be used to alert the user that he has made an error.

The program in Figure 4 demonstrates PERSIST=YES and BLANK.

```
data _null_;        * blank.sas ;
  window whats_up
      #i  @( i*3+i ) 'Where am I?'
        persist=yes
      #21 @20 'Please press '
        'enter' attr=(highlight,blink);
  do i = 1 to 4;
      _msg_='This is with PERSIST=YES.';
      display whats_up;
  end;
  do i = 5 to 8;
      _msg_='The BLANK option clears'||
            ' the PERSISTing data.';
      display whats_up blank;
  end;
  stop;
run;
```

Figure 4

Now that we have seen most of the options of the WINDOW and DISPLAY statements, let's apply this knowledge to manage a "things to do" list. The program in Figure 5 creates a SAS dataset containing test data.

```
data ttd;        * ttd1.sas ;
  input date date7. +1 duration 2. +1
   task $char30.;
  cards;
04jan88  2 complete SASware ballot
08jan88  1 airline reservations for SUGI
08jan88  1 hotel reservations for SUGI
22feb88  1 mail SAS/GRAPH entry
14mar88  1 confirm reservations
15mar88  1 request travel advance
27mar88  4 attend SUGI 13 !
31mar88  2 plan for SUGI 14
  run;
proc print; format date date7.; run;
```

Figure 5

To this dataset we want to add a field "DONE", which will allow us to check-off tasks that have been completed; we also want to calculate and display the ending date for each task ("THRU").

```
data ttd;        * ttd2.sas ;
  window to_do
    #3 @10 done $1. a=rev_video c=yellow
           '(x=done)    '
           'task: ' task $30. c=yellow
      #5 @10 date yymmdd6. c=yellow
           ' through ' thru  yymmdd6.
           protect=yes c=white
           +6 duration 2. ' day(s)'
  ;
  set ttd;
  drop thru;
  thru = date + duration - 1;
  display to_do;
run;
```

Figure 6

New variables DONE and THRU are created in Figure 6 when they first appear in the program, in the window statement. Each variable is followed by a format; this format also must be suitable as an informat, since the same specification is used to display output and to accept input. Formats and informats may also be specified in FORMAT, INFORMAT, and ATTRIB statements. If present, the format in the WINDOW statement takes precedence.

Also used in Figure 6 is PROTECT=YES on field THRU. Since the value of this field is the result of a calculation, we only want the program to be able to change the value, not the user.

"When you display a window containing fields into which you can enter values, you must either enter values or press ENTER at each unprotected field to cause SAS to proceed to the next display. You cannot skip any fields. ... SAS execution proceeds to the next display only after you have pressed ENTER in all unprotected fields." [2] If there were sixty fields on the display, you would have to press ENTER sixty times. Fortunately, there are a few shortcuts; but beware, these may be "bugs" that will go away in a future release -- or they may be "undocumented features". If you press HOME followed by ENTER, the program will advance as if you had pressed enter at each field. The same result can be achieved by just pressing F8, F9, PgUp, or PgDn.

The program in Figure 6 allows us to change the data, but it does not allow us to see the result of any change before it advances to the next record. Figure 7 shows a program that will redisplay the observation (after recalculating the THRU field) until the user types "ok" on the command like. Here we see a new feature: the automatic variables _CMD_ and _MSG_. Both of these variables are automatically created when the WINDOW statement is used. Both are 80 byte character variables, and they are not added to the dataset being created. _CMD_ is cleared before each execution of the DISPLAY statement; _MSG_ is cleared after. When a command is typed on the command line, SAS determines if it is a valid Display Manager (DM) command. If it is, DM executes the command; if not, the command is stored in the _CMD_ variable where the program can examine it and take appropriate action. The line below the command line is where the value of _MSG_ is displayed. _MSG_ can be used to tell the user that he has entered in incorrect value, or to give the user instructions.

```
data ttd;        * ttd3.sas ;
    window to_do
        #3 @10 done $1. a=rev_video
             c=yellow '(x=done)    '
           'task: ' task $30. c=yellow
        #5 @10 date yymmdd6. c=yellow
           ' through ' thru  yymmdd6.
             protect=yes c=white
          +6 duration 2. ' day(s)'
        ;
    set ttd;
    drop thru;
    _msg_=' ';
    do until( upcase(_cmd_)='OK' );
        thru = date + duration - 1;
        display to_do;
        _msg_='Type OK on the command '||
              'line to accept changes.';
    end;
run;
```

Figure 7

The problem with the program in Figure 7 is that the user must acknowledge every observation by typing "ok" on the command line, even if no changes were made. In the next program, Figure 8, only observations that have been changed require the "ok" command; those that have not been changed can be acknowledged by pressing ENTER at each field, or more quickly by using one of the undocumented keys such as F8.

```
data ttd;        * ttd4.sas ;
    window to_do
        #3 @10 done $1. a=rev_video
             c=yellow '(x=done)    '
           'task: ' task $30. c=yellow
        #5 @10 date yymmdd6. c=yellow
           ' through ' thru  yymmdd6.
             protect=yes c=white
          +6 duration 2. ' day(s)'
        ;
    set ttd;
    SAVETASK=TASK;       SAVEDATE=DATE;
    SAVEDUR=DURATION;    SAVEDONE=DONE;
    drop thru SAVETASK SAVEDATE
         SAVEDUR SAVEDONE;
    _msg_=' ';
    do until( upcase(_cmd_)='OK' );
        thru = date + duration - 1;
        display to_do;
        IF SAVETASK=TASK AND SAVEDATE=DATE
         AND SAVEDUR=DURATION AND
         SAVEDONE=DONE THEN _CMD_='OK';
      ELSE _msg_='Type OK on the command'||
                 ' line to accept changes.';
    end;
  run;
```

Figure 8

The changes in Figure 8 are in capital letters. When a new observation is read, the value of each variable is stored. After each time the DISPLAY is executed, the current values are compared to the original values. If all are the same, "ok" is typed for the user (_CMD_ = 'OK'); if any field changed, the user is instructed to type "ok".

---

2. SAS Language Guide for Personal Computers, Version 6 Edition, page 233.

[Three additional programs were used in this part of the workshop presentation, but are omitted here. The features shown are the GROUP= [3,4] specification of the WINDOW statement, multiple windows, and the PERSIST= [4] field option with the DISPLAY statement BLANK [5] option. Refer to the SUGIHILO.SAS program for an example of multiple windows.]

Many of the features discussed in this presentation are new with release 6.03; to get the whole story, you will need both the SAS Language Guide for Personal Computers, Version 6 Edition (pages 117 and 228-238) and SAS Technical Report P-171, Changes and Enhancements to Base SAS Software for Personal Computers, Release 6.03, pages 47 and 72-74.

[The last program in the workshop presentation demonstrated most of the available features. It included menu-driven "help" screens, which consolidated the information in the works cited above. The information was organized in four topics: DISPLAY, WINDOW, automatic variables, and helpful hints. Because of the length of the program, it has been omitted from this paper.]

My thanks to the workshop facilitators:
Cathryn Gust, Klemm Analysis Group, Washington DC
Penni Korb, Economic Research Service USDA, Washington DC
David L. Kuhn, Idea Design, Laguna Niguel CA
Marilyn Sorenson, U.S. House of Representatives, Washington DC
William M. Taylor, DataSoft, Clarksville MD

Comments and questions are welcome. Contact:

Ross Z. Merlin (703) 875-1640

Pinkerton Computer Consultants, Inc.
1900 N. Beauregard Street, Suite 200
Alexandria, VA 22311 (703) 820-5571

---

3. ibid., page 229.
4. SAS Technical Report P-171, Changes and Enhancements to Base SAS Software for Personal Computers, Release 6.03, pp. 72-74.
5. op. cit., SAS Language Guide ..., page 117.