Chapter 930

Macros

Introduction

This software has an interactive (point and click) user interface which makes in easy to learn and use. At times, however, it is necessary to repeat the same steps over and over. When this occurs, a batch system becomes more desirable. This chapter documents a batch language that lets you create a macro (script or program) and then run that macro. With the click of a single button, you can have the program run a series of procedures.

We begin with a discussion of how to create, modify, and run a macro. Next, we list all of the macro commands and their functions.

Macro Command Center

This section describes how to create a macro, edit it, and run it. This is all accomplished from the Macro Command Center window. You can load this window by selecting *Macros Home* from the *Tools* menu.

We will now describe each of the objects on the Macro Command Center window.

Active Macro

This box displays the name of the currently selected macro file. A preview of this macro is displayed in the Preview window. You can change this name by typing over it or by selecting a different macro from the *Existing Macros* box.

Existing Macros

This box displays a list of all existing macros. Click on a macro in this list to make it the active macro—the macro that is used when the Play Macro button to the right is pressed. Double-clicking a macro causes that macro to open in the window's Notepad program for editing.

The macros are stored in the *MACROS* subdirectory of the *PASS 11* folder. They always have the file extension ".ncm".

Preview of Active Macro

This box displays the beginning of the macro that is in the Active Macro box.

Record Macro

Pressing this button causes the commands you issue to be written to the macro file named in the *Active Macro* box. The Macro Command Center window will be replaced by a small *Macro* window that lets you stop recording the macro. When the recording starts, if there is a previous macro file with the same name as that in the Active Macro Name box, the previous macro file is erased.

The ***PASS*** macro recorder does not record every keystroke and click that you make. Instead, it records major operations. For example, suppose you want to include calculating the sample size for a t-test in your macro. You would load the t-test procedure, change some options, and run it. The macro recorder saves a copy of the t-test settings as a template file and writes a single command line to the macro file that references this template. All of your settings are included in the template file—there is no reference in the macro to the individual settings changes. This makes the macro much smaller and easier to modify.

Functions from the View, File, and Edit menus are not recorded during a macro recording. Those functions are performed using the SendKeys commands or other specific commands. As a general rule, the loading, running, and unloading of procedures that have templates are recorded during a macro recording.

Edit Macro

Pressing this button causes the Active Macro to be loaded in the Windows NotePad program. This program lets you modify the macro and then save your changes.

Play Macro

Pressing this button causes the Active Macro to be run. Once the macro is finished, the Macro Command Center window will close and you can view the results of running your macro.

Delete Macro

Pressing this button causes the Active Macro to be deleted. The macro file is actually moved to the Recycle Bin from which it can be rescued if you decide it shouldn't have been deleted.

Close

Pressing this button closes the Macro Command Center window.

Set Menu Macro

A play macro menu item is available from the Home window and the various procedure windows. Clicking this Play menu item causes the designated macro to be run. The macro that is associated with the macro button is controlled by this section of the Macro Command center. To change the macro that is associated with the play macro menu item, simply select the desired macro from the *Existing Macros* list and then click the Menu Macro button. This will associate the macro to the menu item. This association will remain even if the program is exited.

Syntax of a Macro Command Line

PASS macros are line based. That is, each macro command expression is written on a separate line. The basic structure of a line is that it begins with a *command* followed by one or more options or parameters of the command, called *arguments*. For example, the following macro loads a procedure, loads a template for that procedure, and then runs the procedure with the loaded template.

```
LoadProc PSProp2IP
LoadTemplate PSProp2IP "Template1"
RunProc PSProp2IP
```

In this example, LoadProc, LoadTemplate, and RunProc are commands, while PSProp2IP and "Template1" are arguments.

Comment Lines

It is often useful to add comment lines to a macro to make it easier to understand later. Comment lines begin with single quotes. When the macro processor encounters a single quote at the beginning of a line, the rest of the line is ignored. Single quotes occurring at a location other than the beginning of a line are treated as text.

Blank lines may also be added to a macro to improve readability. These are also ignored.

Macro Constants and Macro Variables

As stated above, macro command lines consist of keywords followed by arguments. These arguments are either constants, such as "Template1" or 'PSProp2IP', or macro variables. These will be discussed next.

Macro Constants

Macro constants are fixed values. There are two types of macro constants: text and numeric.

Numeric Constants

Numeric constants are numbers. They may be whole or decimal numbers. They may be positive or negative. They may be enclosed in double quotes, although this is not necessary. When the macro processor expects a number but receives a text value, it sets the numeric value to zero.

Examples of numeric constants are

```
1, 3.14159, and 0.
```

Text Constants

Text constants are usually enclosed in double quotes. If a constant is a single word (made of letters and digits with no blanks or special characters), the double quotes are not necessary.

Examples of text constants are

```
Apple, "Apple Pie", and "D:/Program Files".
```

Macro Variables

Macro Variables are used to store temporary values for use in macro command lines. Some examples of assigning values to macro variables are

```
A# = 4
B# = 4 + 3
File$ = "C:/Program Files/PASS/Data/ABC.S0"
F$ = "4" & "5"
```

In these examples, A#, B#, File$, and F$ are macro variables. The assigned values for each of the variables are 4, 7, "C:\Program Files\PASS\Data\ABC.S0", and 45, respectively.

There are two types of macro variables: text and numeric.

Text Macro Variables

Text macro variables are used to hold text values. The rules for naming them are that the names can contain only letters and numbers (no spaces or special characters) and they must end with a '$'. The case of the letters is ignored (so 'A$' is used interchangeably with 'a$').

Examples of text macro variable names are

```
A$
Apple$
FileName$
```

Numeric Macro Variables

Numeric variables are used to hold numeric values. The rules for naming them are that the names can contain only letters and numbers (no spaces or special characters), and they must end with a '#'. The case of the letters is ignored (so 'A#' is used interchangeably with 'a#').

Examples of numeric macro variable names are

```
A#
Apple#
NRows#
```

Assigning Values to Macro Variables

One type of macro variable expression is that of assigning a value to a variable. The basic syntax for this type of expression is

{variable} = {value}

where the {variable} is text or numeric and {value} is a macro variable or (text or numeric) macro constant. If a text value contains spaces or special characters, it must be enclosed in double quotes.

A text value can be is assigned to a numeric macro variable. In this case, the text value is converted to a number. If it cannot be converted to a number (e.g., it is a letter), the numeric macro variable is set to zero.

Following are some examples of valid assignment expressions.

```
A# = 4
X$ = John
File$ = "C:\Program Files\PASS\Data\ABC.S0"
X$ = File$
X$ = A#
A# = F$
F$ = 4
```

Macro Variable Combination Expressions

Macro variables can be combined using simple mathematical expressions. The basic syntax for this type of expression is

{variable} = {value} {operator} {value}.

The available operators are + (add), - (subtract), * (multiply), / (divide), and & (concatenate).

If a text value is involved in a mathematical expression, it is converted to a numeric value before the mathematical expression is evaluated. If it cannot be converted to a number, the text value is set to the numeric value of zero.

Following are some examples of valid assignment expressions.

Expression	Result
A# = 4 + 3	A# = 7
B# = A# * 2	B# = 14
C$ = "C:/Pgm/"	
D$ = C$ & "ABC.S0"	D$ = "C:/Pgm/ABC.S0"
E# = C$ * 4	E# = 0
F$ = "4" + "5"	F$ = "9"
F$ = "4" & "5"	F$ = "45"
A# = 1	
A# = A# + 1	A# = 2

Macro variable assignments are used while the macro is running, but are not saved when the macro has completed.

Displaying Macro Variables

The value of one or more macro variables may be displayed on the output using the PRINT or HEADING commands.

Print

This command outputs the requested values to the printout.

The syntax of this command is

PRINT {p1} {p2} {p3} …

where

{p1} {p2} ... are assigned macro variables or constants.

Following are some examples of Print commands.

Command	Printed Result
PRINT "Hi World"	Hi World
I# = 1	
J$ = "C:/PASS/Data/ABC"	
F$ = J$ & i#	
F$ = F$ & ".s0"	
PRINT "File=" F$	File=C:/PASS/Data/ABC1.s0
PRINT "1" "2" "3" "4"	1 2 3 4

Heading

This command adds a line to the page heading that is shown at the top of each page.

The syntax of this command is

HEADING {h1}

where

{h1} is an assigned macro variable or constant.

Following are some examples of valid HEADING commands.

Command	Heading
HEADING "Hi World"	Hi World
F$ = "Heart Study"	
HEADING F$	Heart Study

Logic and Control Commands

The lines in a macro are processed in succession. These commands allow you to alter the order in which macro lines are processed, allow user-input, or end the program.

List of Logic and Control Commands:

Flag
GOTO
IF
INPUT
END
SendKeys

Flag Statement

A flag is a reference point in the program. The GOTO command sends macro line control to a specific flag. A flag is made up of letters and numbers (no spaces) followed by a colon.

Following are some examples of valid flags. More extensive examples are shown in the description of the IF statement (below)

Examples
```
Flag1:
A:
Loop1:
```

GOTO command

This command transfers macro processing to the next statement after a flag.

The syntax of this command is

GOTO {P1}

where

{P1} is a text variable or text constant.

Following are some examples of valid GOTO commands.

Examples
```
GOTO Flag1
or
F$ = "Flag1"
GOTO F$
```

IF command

This command transfers macro processing to the next statement after a flag if a condition is met. The syntax of this command is

IF {p1} {logic} {p2} GOTO {p3}

where

{p1} is a variable or constant.

{p2} is a variable or constant.

{p3} is a flag.

{logic} is a logic operator. Possible logic operators are =, <, >, <=, >=, and <>.

Following are some examples of valid IF commands.

Examples
```
IF x1# > 5 GOTO flag1
IF y$ = "A" GOTO flag2
IF y$ <> "A" GOTO flag3
```

INPUT

This command stops macro execution, display a message window, and waits for a value to be input. This value is then stored in the indicated macro variable.

The syntax of this command is

INPUT {variable} {prompt} {title} {default}

where

{variable} is the name of the variable (text or numeric) to receive the value that is input.

{prompt} is the text phrase that is shown on the input window.

{title} is the text phrase that is displayed at the top of the input window.

{default} is the default value for the input.

Following is an example of this command.

Example
```
INPUT A# "Enter the number of items" "Macro Input Window" 1
```

END

This command closes the ***PASS*** system.

The syntax of this command is

END

Example
```
END
```

SendKeys

This command sends one or more keystrokes to the program as if you had typed them in from the keyboard. This facility allows you to create macros to accomplish almost anything you can do interactively within the program.

To use this, run the program from the keyboard, noting exactly which keys are pressed. Then, type the appropriate commands into the sendkeys text. Note that spaces are treated as characters, so '{down} {tab}' is different from '{down}{tab}'.

The syntax of this command is

SendKeys {value}

where *{value}* is a text constant or variable.

Remarks

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use "A" for value. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for string.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use {+}. Brackets ([ ]) have no special meaning to SendKeys, but you must enclose them in braces. To specify brace characters, use {{} and {}}.

To specify characters that are not displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the following codes:

Key	Code
Backspace	{bs}
Break	{break}
Caps Lock	{capslock}
Delete	{delete}
Down Arrow	{down}
End	{end}
Enter	{enter}
Esc	{esc}
Home	{home}
Insert	{insert}

Macros

Key	Code
Left Arrow	{left}
Num Lock	{numlock}
Page Down	{pgdn}
Page Up	{pgup}
Right Arrow	{right}
Tab	{tab}
Up Arrow	{up}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

Key	Code
Shift	+
Ctrl	^
Alt	%

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 4} means press the LEFT ARROW key 4 times; {h 8} means press H 8 times.

Spreadsheet Note: When a macro is run the usual beginning location on the screen is the new page icon (just below File) in the upper left of the screen. A single tab may be entered in the macro to go to the upper left cell position on the spreadsheet.

Examples	Action
SendKeys "ABC {enter}"	(Types ABC and then enter)
SendKeys "{enter right}"	(Enter and then down arrow)
SendKeys "%H{down 2}{enter}"	(Activates the Serial Numbers from the Help menus)
SendKeys "{right}"	(Moves to the right one cell)

Window Position Commands

These commands allow the user to position or hide windows while the macro is running.

List of Window Commands:

WindowLeft
WindowTop

WindowLeft

This command sets the position of the left edge of the home, panel, and output windows. This allows you to effectively hide these windows while a macro is running.

The syntax of this command is

WINDOWLEFT {value}

where

{value} is the value of the left edge in thousandths of an inch.

Examples	Action
WINDOWLEFT 0	(positions the left edge to zero)
WINDOWLEFT 10000	(positions the left edge ten inches to the right)
WINDOWLEFT -10000	(positions the left edge ten inches to the left)

WindowTop

This command sets the position of the top of the home, panel, and output windows. This allows you to effectively hide these windows while a macro is running.

The syntax of this command is

WINDOWTOP {value}

where

{value} is the value of the left edge in thousandths of an inch.

Examples	Action
WINDOWTOP 0	(positions the top to zero)
WINDOWTOP -10000	(positions the top ten inches up)

Spreadsheet Commands

The following commands open, close, and modify an ***PASS*** spreadsheet.

List of Dataset Commands:

GetCell
SetCell
NumRows
GetMaxRows
GetMaxCols

GetCell

This command obtains the value of a spreadsheet cell. The syntax of this command is

GetCell {procedure} {column} {row} {macro variable}

where

{procedure} is the macro name or number of the procedure that uses a spreadsheet. For the output spreadsheet, the number is 73 and the name is Output.

{column} is the name or number of the column with the cell to be read.

{row} is the row number of the cell to be read.

{macro variable} is the text or numeric macro variable that will hold the value of the spreadsheet cell.

Examples
```
GetCell Output "HeartRate" 27 H#
GetCell Output Name 16 Name16$
```

SetCell

This command sets a spreadsheet cell to a specified value. The syntax of this command is

SetCell {procedure} {col1} {row1} {row2} {value}

where

{procedure} is the macro name or number of the procedure that uses a spreadsheet. For the output spreadsheet, the number is 73 and the name is Output.

{col1} is the name or number of the column to receive the new value.

{row1} is the first row in a range of rows to receive the new value.

{row2} is the last row in a range of rows to receive the new value

{value} is the new value. This value may be text or numeric.

Examples
```
SetCell Output "HeartRate" 10 10 "100"
SetCell Output 1 10 20 100
```

NumRows

This command loads the number of rows used in a specific column into a macro variable. The syntax of this command is

NumRows {procedure} {col1} {n}

where

{procedure} is the macro name or number of the procedure that uses a spreadsheet. For the output spreadsheet, the number is 73 and the name is Output.

{col1} is a column name or number on the current spreadsheet.

{n} is a numeric macro variable.

Examples
```
NumRows Output "HeartRate" n1#
NumRows Output 1 n#
```

GetMaxRows

This command loads the maximum number of rows used by any column into a macro variable.

The syntax of this command is

GetMaxRows {procedure} {n}

where

{procedure} is the macro name or number of the procedure that uses a spreadsheet. For the output spreadsheet, the number is 73 and the name is Output.

{n} is a numeric macro variable.

Examples
GetMaxRows Output n1#
GetMaxRows Output n#

GetMaxCols

This command causes the number of the right-most column with data to be loaded into a macro variable.

The syntax of this command is

GetMaxCols {procedure} {n}

where

{procedure} is the macro name or number of the procedure that uses a spreadsheet. For the output spreadsheet, the number is 73 and the name is Output.

{n} is a numeric macro variable.

Example
GetMaxCols Output nvars#

Procedure Commands

The following commands open, modify, run and close procedures.

List of Procedure Commands:

LoadProc
RunProc
SaveTemplate
UnloadProc
Option

LoadProc

This command loads the designated procedure window. Once loaded, the options of the procedure may be modified and then the procedure can be executed.

The syntax of this command is

LoadProc {proc} {template}

where

{proc} is a variable or constant that gives the name or number of the procedure to be loaded. Each procedure's name and number is displayed near the bottom of the window under the Template tab.

{template} is an optional text variable or text constant that gives the name of a template file that is loaded with this procedure. If this value is omitted, the default (last) template for this procedure is loaded. Note that the text value does not include the extension or the folder information for the template file.

Following are some examples of valid LOADPROC commands.

```
Example
LOADPROC 24 "macro 1"
LOADPROC PSProp2IP "macro 1"
LOADPROC PSProp2IP
LOADPROC 24
```

RunProc

This command executes the indicated procedure. The syntax of this command is

RunProc {proc} {template}

where

{proc} is a variable or constant that gives the name or number of the procedure to be run. Each procedure's name and number is displayed near the bottom of the window under the Template tab.

{template} is a required variable or text constant that gives the name of the resulting template file. Note that the text value does not include the extension or the folder information for the template file.

SaveTemplate

This command saves the settings in the last procedure loaded to a template file. Once loaded, the options of the procedure may be modified and then the procedure can be executed.

The syntax of this command is

SaveTemplate {proc} {template} {id}

where

{proc} is a variable or constant that gives the name or number of the procedure whose template is to be saved. Each procedure's name and number is displayed near the bottom of the window under the Template tab.

{template} is a required variable or text constant that gives the name of the resulting template file. Note that the text value does not include the extension or the folder information for the template file.

{id} is an optional text variable or text constant that is stored with the file. This text is displayed with the file name under the Template tab.

Following are some examples of valid SaveTemplate commands.

```
Example
SaveTemplate PSProp2IP "Template1"
SaveTemplate PSProp2IP "Template1" "This template was created by a macro on January 1"
```

UnloadProc

This command closes the indicated procedure window. The syntax of this command is

UnloadProc {proc}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom of the window under the Template tab.

Option

This command lets you set the values of the individual options of a procedure. If the option numbers are not displayed on the procedure windows, go to View or File, Options, View, and check the box next to 'Show Option Numbers'.

For example, you may want to change the value of Alpha or the type of test that is run.

The syntax of this command is

Option {proc} {number} {value1} {value2} {value3} …

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom of the window under the Template tab.

{number} is the number of the option that is to be set. This number is displayed at the lower, left corner of the procedure window when the mouse is positioned over that option.

If the 'Opt' value is not displayed, activate it by doing the following: from the spreadsheet menus select File or View, Options, and View. Check the option labeled 'Show Option Numbers'.

{value1} is the new value of the option.

{value2}… are the new values of the remaining parameters of the option. Most options only have one value, so a second value is not necessary. However, a few options, such as text properties, bring up a window for option selection. These options have two or more parameters.

Following are some examples of valid OPTION commands.

Example
```
OPTION 24 2 4
OPTION "PSProp2IP" 3 0.8
```

Output Commands

The following commands manage the output (word processor) windows.

List of Output Commands:

SaveOutput
ClearOutput
PrintOutput
AddToLog
NewLog
SaveLog
OpenLog

SaveOutput

This command saves the current output to the designated file name.

The syntax of this command is

SaveOutput {filename}

{filename} a text constant or variable that gives the name of the file to receive the output. Note that the extension of the file name should be '.RTF'.

Example
```
SAVEOUTPUT "C:/Program Files/PASS/Sample.rtf"
```

ClearOutput

This command clears (erases) the current output.

The syntax of this command is

ClearOutput

Example
CLEAROUTPUT

PrintOutput

This command prints the current output.

The syntax of this command is

PrintOutput

Example
PRINTOUTPUT

AddToLog

This command copies the output in the output window to the log window. Note that nothing is saved by this command.

The syntax of this command is

AddToLog

Example
ADDTOLOG

NewLog

This command clears log window.

The syntax of this command is

NewLog

Example
NEWLOG

SaveLog

This command saves the current contents of the log output window to the designated file name.

The syntax of this command is

SaveLog {filename}

{filename} a text constant or variable that gives the name of the file to receive the log. Note that the extension of the file name should be '.RTF'.

Example
```
SAVELOG "C:/Program Files/PASS/Reports/Sample.rtf"
```

OpenLog

This command opens and displays the contents of the specified file.

The syntax of this command is

OpenLog {filename}

{filename} a text constant or variable that gives the name of the file to opened. Note that only RTF files can be opened.

Example
```
OPENLOG "C:/Program Files/PASS/Sample.rtf"
```

Alphabetical Macro Command List

AddToLog
ClearOutput
End
GetCell {proc} {variable} {row} {macro variable}
GetMaxCols {proc} {n}
GetMaxRows {proc} {n}
Goto {P1}
Heading {h1}
If {p1} {logic} {p2} goto {p3}
Input {variable} {prompt} {title} {default}
LoadProc {proc} {template}
LoadTemplate {proc} {template}
NewLog
NumRows {proc} {v1} {n}
OpenLog {filename}
Option {number} {value1} {value2} {value3} …
Print {p1} {p2} {p3} …
PrintOutput
RunProc {proc}
SaveLog {filename}
SaveOutput {filename}
SaveTemplate {proc} {template} {id}
SendKeys {value}
SetCell {proc} {v1} {row1} {row2} {value}
UnloadProc {proc}
WindowLeft {value}
WindowTop {value}

Examples

The following section provides examples of PASS macros. Our intention is that these examples will help you learn how to write macros to accomplish various repetitive tasks with PASS.

Example 1 – Automatically Run a Procedure

This macro opens the two-sample T-test procedure and runs the procedure with the default values.

```
'***  Load the Two-Sample T-Test procedure
'***  Note that the default values are used.
LoadProc PSMean2ID

'***  Run the analysis
RunProc PSMean2ID
'*** End of Macro 1
```

Example 2 – Run Several Procedures

This macro runs three procedures. The two-sample T-test procedure is run with the default values, except that Alpha is set to 0.10. The one-sample T-test procedure is run with default values, except that Alpha is set to 0.15. The linear regression procedure is run with the values set by the template ‘My LinReg Template’.

```
'***  Load the Two-Sample T-Test procedure
LoadProc PSMean2ID
'***  Set Alpha (Option 8) to 0.10
Option PSMean2ID 8 0.10
'***  Run the analysis
RunProc PSMean2ID

'***  Load the One-Sample T-Test procedure
LoadProc PSMeanCorr
'***  Set Alpha (Option 8) to 0.15
Option PSMeanCorr 8 0.15
'***  Run the analysis
RunProc PSMeanCorr

'***  Load the Linear Regression procedure
LoadProc PSLinReg
'***  Load the ‘My LinReg Template’ template
LoadTemplate PSLinReg "My LinReg Template"
'***  Run the analysis
RunProc PSLinReg
```