30
Controlling Macro Flow

- **IF, ELSEIF, ELSE, ENDIF**, 30-2
- **DO, ENDDO**, 30-3
- **WHILE, ENDWHILE**, 30-4
- **NEXT**, 30-5
- **BREAK**, 30-6
- **GOTO, MLABEL**, 30-6
- Invoking Macros from Within Macros, 30-7
- **CALL, RETURN**, 30-8
- **EXIT**, 30-10
- **PAUSE, RESUME**, 30-10
- Using DOS Commands, 30-11

To Table of Contents  To Index
Chapter 30 IF, ELSEIF, ELSE, ENDIF

Control statements allow you to control the order in which commands in the macro are executed. You can nest control statements. For example, an IF can contain another IF, which can contain a DO, and so on.

**IF, ELSEIF, ELSE, ENDIF**

IF logical expression
   (a block of MINITAB and macro commands)
ELSEIF logical expression
   (a block of MINITAB commands and macro statements)
ELSE
   (a block of MINITAB commands and macro statements)
ENDIF

Allows you to execute different blocks of code depending on a logical condition. A logical expression is any expression from the LET command. The comparison and Boolean operators listed below are the features of LET that are most often used in IF.

- = or EQ equal to
- ≠ or NE not equal to
- < or LT less than
- > or GT greater than
- ≥ or GE greater than or equal to
- ≤ or LE less than or equal to
- ~ or NOT
- & or AND
- | or OR

In most cases the logical expression evaluates to a single number. If the number is 0 (false), the block of statements is skipped; if it is not 0 (true), the block is executed. If the logical expression evaluates to a column, then if all entries in the column are 0, the expression is considered false, otherwise it is considered true.

You can use up to 50 ELSEIF statements within the IF-ENDIF block.
Here is a simple example, using a global macro:

```
GMACRO
SMALL
#
# Takes the data in C1-C3. Finds the column with the smallest mean
# and prints it out. If, because of ties, there is no single column
# with the smallest mean, a message is printed.
#
LET K1 = MEAN(C1)
LET K2 = MEAN(C2)
LET K3 = MEAN(C3)
IF  K1 < K2  AND  K1 < K3
   PRINT  C1
ELSEIF  K2 < K1  AND  K2 < K3
   PRINT  C2
ELSEIF  K3 < K1  AND  K3 < K2
   PRINT  C3
ELSE
   NOTE Note: There are ties.
ENDIF
ENDMACRO
```

**DO, ENDDO**

```
DO      K = list of numbers
      (a block of MINITAB commands and macro statements)
ENDDO
```

DO, ENDDO allows you to loop through a block of commands. K is set equal to the first number in the list, then the block of commands is executed. When MINITAB reaches the ENDDO, K is set equal to the next number in the list and the block is executed again. This continues until all numbers in the list are used, or until you branch out of the DO-loop with a BREAK, GOTO, RETURN, or EXIT command.

The list of numbers can be an explicit list of any numbers or stored constants. A patterned list can be abbreviated using a colon and slash as in SET. For example, 1:10 is the list 1, 2, 3, …, 10, and 1:1.8 / .2 is the list 1, 1.2, 1.4, 1.6, 1.8. Numbers can be increasing or decreasing order. The following DO-loop changes the values in rows 1 through 10 and row 50 of columns C1 and C2 to the missing value code:

```
DO  K1 = 1:10 50
   LET C1(K1) = '*'
   LET C2(K1) = '*'
ENDDO
```
Here is a local macro that calculates a moving average of length three. It shows how to loop through the values in a column.

```
MACRO
MOVAVE X Y
#
# Calculates the simple moving average of the data in X and stores the answer in Y.
#
MCONSTANT N I
MCOLUMN X Y
LET N = COUNT(X)
LET Y(1) = '*'
LET Y(2) = '*'
DO I = 3 : N
   LET Y(I) = (X(I) + X(I-1) + X(I-2))/3
ENDDO
ENDMACRO
```

**WHILE, ENDWHILE**

Repeats a block of commands as long as the logical expression is true. The logical expression follows the same rules as in the IF statement.

Suppose we want to find the root of the equation, $y = -1 + x + x^3$. We know this equation has just one real root and that it is between 0 and 1. The following global macro allows us to find, approximately, what the root is.

```
GMACRO
ROOT
#
# Finds the root of a specific polynomial. The result is within .01 of the exact answer.
# K90-K93 are used for scratch work
#
NAME K90 = 'X'  K91 = 'Y'  K92 = 'Xlow'  K93 = 'Ylow'
LET 'X' = 0
LET 'Y' = -1
WHILE 'Y' < 0
   LET 'X' = 'X' + .01
   LET 'Y' = -1 + 'X' + 'X'**3
ENDWHILE
LET 'Xlow' = 'X' - .01
LET 'Ylow' = -1 + 'Xlow' + 'Xlow'**3
PRINT 'Xlow' 'Ylow'  'X' 'Y'
ENDMACRO
```
We first initialized the two variables, X and Y, to 0 and −1. Each time through the
WHILE-loop, MINITAB first checks to see that Y is still less than zero. If it is, we
increase X by .01 and calculate Y at this new value. Once the condition fails, that is,
once Y is no longer less than zero, we exit the loop and go to the first statement after
ENDWHILE. Then we print out the answer.

NEXT

Transfers control from within a DO- or WHILE-loop back to the beginning of the
block. For DO, the loop variable is then set to the next value in the list and the loop is
executed again. Here is a simple example, using a global macro.

**GMACRO**
**FIVES**

# Takes the column named X and changes all entries
# that are greater than 5 to 5.
# Constants K90 and K91 are used for scratch work.

**NAME** K90 = 'N'  K91 = 'I'
**LET** 'N' = COUNT('X')
**DO** 'I' = 1 : 'N'
   **IF** 'X'('I') <= 5
      **NEXT**
   **ELSE**
      **LET** 'X'('I') = 5
      **ENDIF**
   **ENDDO**
**ENDDO**
**ENDMACRO**

The DO-loop goes through all the values in X. If a value is less than or equal to 5,
NEXT passes control to the top of the DO-loop and the value is left unchanged. If a
value is greater than 5, the ELSEIF block is executed and that value is set to 5.
**BREAK**

Transfers control from within a DO- or WHILE-loop to the command immediately following the end of the loop. Thus BREAK breaks out of the loop. Here is a simple example using a global macro.

```plaintext
GMACRO
NOMISS
#
# Takes data from the column named X. Finds the first missing # observation. Then deletes all observations starting with the # first missing to the end of the column.
# Constants K90 and K91 are used for scratch work
#
LET K90 = COUNT('X')
DO K91 = 1:K90
  IF 'X'(K91) = '*'
    BREAK
  ENDIF
ENDDO
DELETE K91:K90 'X'
ENDMACRO
```

The program goes through the values of X until it finds a missing value. It then leaves the loop and goes to the statement following ENDDO—in this example, DELETE.

**Note**

This program does not handle the case when X has no missing values correctly. We will fix this when we discuss the command EXIT.

---

**GOTO, MLSLABEL**

```
GOTO number
   (other MINITAB commands and macro statements)
MLABEL number
```

Allows you to branch to any line in your macro. There can be several GOTOs in one program. A GOTO is matched to the MLSLABEL that has the same number. The number can be any integer from 1 to 8 digits long. It cannot be a variable.
Here is the program we used to illustrate BREAK above, but now coded with a GOTO.

```
LET K90 = COUNT('X')
DO K91 = 1 : K90
  IF 'X'(K91) = '*' GOTO 5
ENDDO
MLABEL 5
DELETE K91:K90 'X'
```

**Invoking Macros from Within Macros**

You may have two or more macros in one file. Each macro in the file follows the usual structure (beginning with GMACRO or MACRO, ending with ENDMACRO, etc.), and each must have a unique template name. When you invoke a macro, MINITAB executes the first macro in the file. Subsequent macros in the file are subroutines that you can invoke using a CALL statement (see CALL, RETURN on page 30-8).

There are some restrictions on which type of macro another macro can call:

<table>
<thead>
<tr>
<th>From within this type of macro</th>
<th>You can invoke...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>Global Exec</td>
</tr>
<tr>
<td>Local</td>
<td>Local Exec</td>
</tr>
<tr>
<td>Exec</td>
<td>Global Exec</td>
</tr>
</tbody>
</table>

You invoke a macro from within a macro in the same way you invoke a macro from the MINITAB prompt. On a line, put the symbol % followed by the name of the macro file, as in %TRIM. You can also include a path statement, as in %C:\MYWORK\TRIM. If it is a local macro, include all appropriate arguments and subcommands.

Because the macros you execute are stored in your worksheet area, the only limitation to the number of macros you can nest is the amount of space available in your worksheet. If you run out of room, see the macro statement SWAP (page 32-6) for a way to work around that problem.

The following example improves the global macro ANALYZE, described in Chapter 28, to handle the case when a data set is too small to analyze. The main file, stored as ANALYZE2.MAC, determines how many observations are in the data set. If there are fewer than 5, it invokes the macro file TOOSMALL.MAC. TOOSMALL prints out a message then prints the data set. If the data set has at least 5 observations, ANALYZE2 invokes the macro file OK.MAC. OK is the same as the original version, ANALYZE.
CALL, RETURN

You can include several macros in one file, just as a program often includes several subroutines. CALL and RETURN let you specify when to pass control to another macro and when to return to the main macro. You can include several global macros in one file, or several local macros in one file, but you cannot mix global and local macros together in one file.

When you invoke a macro, from interactive MINITAB or from another macro, the first macro in the file is executed first. Use the macro statements CALL and RETURN to invoke a different macro within the macro file.
CALL, RETURN

Recall that the second line of a macro is the template, or the macro name. When one macro in a macro file calls another macro in that file, use the command CALL, followed by the name on that macro's template. If it is a local macro, include appropriate arguments and subcommands. Any macro in a macro file can CALL any other macro in the file, any number of times.

RETURN says to leave the current macro and go back to the calling macro, to the statement just after the CALL. RETURN is optional. If RETURN is not present in the macro that was called (the subroutine), then after it has executed, control is transferred back to the calling macro.

The following example is a variation on ANALYZE2.MAC (page 30-7) named ANALYZE3.MAC. This global macro file contains three macros:

```
GMACRO
ANALYZE3
#
NOTE Would you like all data printed?
YESNO K80
#if user types "yes" K80 = 1, if "no" K80 = 0
LET K90 = COUNT(C1)
 IF K90 < 5
   CALL TOOSMALL
 ELSE
   CALL OK
 END IF
#
#if
 IF K80 = 1
 NOTE Here are the data.
 PRINT C1-C3
 END IF
ENDMACRO
#
#
GMACRO
TOOSMALL
NOTE Data set has fewer than 5 observations.
 NOTE No analysis will be done.
ENDMACRO
#
#
GMACRO
OK
 NAME C1 = 'Yield'  C2 = 'Chem1'  C3 = 'Chem2'  C5 = 'Ln. Yield'
 DESCRIBE C1-C3
 LET C5 = LOGE( 'Yield' )
 REGRESS C5 2 C2 C3
 IF K80 = 1
   RETURN
 END IF
 NOTE Analysis done, but no data printed by request
ENDMACRO
```

MINITAB User's Guide 1 30-9
In this example, ANALYZE3, we use the YESNO command (see page 31-5) to see if the user wants to print all the data. If the response is “Yes,” YESNO sets K80 to 1; if the answer is “No,” K80 is set to 0.

The OK subroutine checks the value of K80 with an IF statement. If K80 equals 1, the RETURN statement sends control back to the main macro. If K80 is anything else, the macro prints one more note.

When the ENDMACRO statement is encountered in either the TOOSMALL or OK subroutine, control is transferred back to the calling macro.

**EXIT**

EXIT

Stops the macro and transfers control back to interactive MINITAB.

Here is a modification of the macro NOMISS, that correctly handles the case when X contains no missing values.

```
LET K90 = COUNT('X')
DO K91 = 1:K90
  IF 'X'(K91) = '*' BREAK
END F
IF K91 = K90  NOTE Note: There are no missing observations in X
  EXIT
END F
ENDDO
DELETE K91:K90 'X'
```

**PAUSE, RESUME**

When MINITAB encounters a PAUSE in a macro, control is shifted from the macro to the keyboard. You can then type any MINITAB command. When you want to return control to the macro, type RESUME (or just R). PAUSE can help you debug a macro you are developing. It can also allow you to get input from the macro user.
If you are in PAUSE mode from within a local macro, you have access to the local worksheet and only the local worksheet. You can also declare new local variables and use them. They will be stored at the end of the local worksheet.

When you are in PAUSE mode, you can type any MINITAB command. You cannot CALL other macros in the same file, invoke a macro from another macro file, or use control statements.

### Using DOS Commands

#### Change and show directories

<table>
<thead>
<tr>
<th>CD [path]</th>
<th>displays your current directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR [path]</td>
<td>lists the names of files in your current directory</td>
</tr>
</tbody>
</table>

CD without a path displays your current directory. CD with a path changes from your current directory to the specified directory.

DIR lists the names of all the files in your current or the specified directory.

Here are some examples:

- `CD` displays your current directory
- `CD \SUE\SALES91` changes to the \SUE\SALES91 directory
- `DIR` lists the names of files in your current directory

#### Show a text file

`TYPE "[path]filename.ext"`

Lists the specified text (ASCII) file in the Session window.

The file must be a standard text file in order for this to work. Include the full file name and file extension within single quotation marks. If the file is not in your default directory, include the path within single quotation marks as well. For example, to list the contents of the macro file SALES.MAC on your screen, enter:

```
TYPE "SALES.MAC"
```