

## 3 Using Informational Prompts

In this lesson: PROMPT, MESSAGEBOX, MacroStatusPrompt, PauseKey, PAUSE, PauseSet, PauseCommand; MacroPause, WAIT, BEEP, ENDPROMPT

In many macros, you'll want to post messages to inform the user of the macro's progress, or to prompt for some text or other information. The commands in this lesson show you how to display messages, then wait for user interaction, or just pause for a specified length of time.

*Note: I have used different parameter names on many of the commands covered in this book. I have done this for better consistency between commands, to simplify, and to make what each parameter is for more understandable. If you need the PerfectScript parameter names, you can find them in the Command Inserter or in the Online Macros Help.*

### PROMPT

**Syntax:** `PROMPT("Title"; <"Prompt">; <Style>; <HorizontalPosition>; <VerticalPosition>)`

**Purpose:** PROMPT displays your message in a dialog box with an OK and a Cancel button. The entire PROMPT command can be up to 512 characters in length, including the *Title*, *Prompt*, *Style*, and position of the prompt.

**Return:** None

#### Parameters:

The PROMPT command has five parameters that allow you to specify the text to be displayed, the location of the PROMPT dialog on the screen, and whether an icon will be included on the dialog. Remember that the entire PROMPT command can be up to 512 characters in length. Thus, if you do not use the *Style*, *HorizontalPosition* and *VerticalPosition* parameters, the message can be longer. Also remember that you cannot place a hard return in the middle of a parameter. A parameter can wrap to the next line with a soft return, however.

The text you include in the prompt message must be enclosed in quotation marks. You can also display text stored in variables in a prompt message. This allows you to display longer messages than would otherwise be allowed because of the 512 character limit for a macro command. Variable names should not be enclosed in quotation marks as text would be.

You can combine variables with text in the message by including the text in quotes, and following the quote mark with a plus (+) symbol. Next, type the variable name. To add additional text after the variable, add another plus symbol, a quotation mark, then the text. Make sure any spaces you need between words are included between the quote marks.

**Title.** The *Title* parameter contains a title to be displayed in the title bar. Oddly, if you specify a *Title* but not a *Message*, the *Title* will be treated as the message and be displayed in the main body of the PROMPT dialog.






You can specify no title by placing "" (two quotation marks) in the *Title* parameter.

**Prompt.** The *Message* parameter is optional, but this parameter is usually the one that you'll use the most. Your message can be several lines long, and contain any information you want the user to see. If your

message is more than one line long in the dialog box, WordPerfect will wrap it automatically to the next line.

**Style:** The *Style* parameter is optional. This parameter allows you to display one of four predefined icons. When you want an icon to appear in your PROMPT, use one of the values below to include the icon. Example 3-3 shows a PROMPT with the QuestionMark icon.

The following icons are available for display in a PROMPT message.

- |   |                                   |   |
|---|-----------------------------------|---|
| 0 | NoIcon! (default)                 |   |
| 1 | StopSign! or ErrorIcon!           |    |
| 2 | QuestionMark!                     |    |
| 3 | ExclamationPoint! or WarningIcon! |  |
| 4 | InformationIcon! or Asterisk!     |  |
| 5 | ApplicationIcon!                  |  |
| 6 | WindowsLogo!                      |  |

Other available styles are:

- |    |            |   |
|----|------------|---|
| 0  | NoPause!   | PROMPT will not automatically pause (default)   |
| 0  | NoBeep     | PROMPT will not automatically beep (default)  |
| 0  | Buttons!   | PROMPT will automatically have OK and Cancel buttons (default)  |
| 8  | Pause!     | PROMPT command pauses until button is clicked (same as following PROMPT command with a PAUSE command).  |
| 16 | Beep!      | The default beep sounds when message is displayed.  |
| 32 | NoButtons! | Shows prompt without OK and Cancel buttons. (Use a WAIT command or some looping subroutine with this style. This style is very useful when you want a message box to appear that contains an updated status message that the user need not respond to.) |

**HorizontalPosition and VerticalPosition:** The *HorizontalPosition* and *VerticalPosition* parameters are optional. The default position of the PROMPT message is in the middle of the screen. If you include values for left and top positioning, the upper left corner of the PROMPT message is supposed to be placed at the coordinates you set. For instance, setting the coordinates to zero and zero *should* place the upper left corner of the PROMPT in the upper left corner of the screen. However, WordPerfect 8 and 9 seem to have a bug that causes it to calculate incorrect coordinates when you use these parameters. You'll need to experiment with positioning a PROMPT to get the dialog in the correct position. The values you use are based on your screen resolution, so the same values may give you differing results on different computer systems.

### Remarks:

A PROMPT dialog is *modeless*. This means that while a macro PROMPT is displayed on the screen, the user can click in the document (the PROMPT dialog gets focus when it is displayed), type text, or perform other functions while the PROMPT message remains on the screen. The user can also drag the prompt dialog to another location on the screen. When used with a PAUSE or PauseKey command, the PROMPT can instruct the user to add text to the document or perform some other task. The message will remain on the screen until the user presses (Enter), or clicks the OK or Cancel buttons.

*Note: If you click in the document while the PROMPT dialog is on the screen, the WordPerfect window will come forward, covering the PROMPT dialog. Press (Alt+Tab) to get the PROMPT dialog back.*

When the user presses (Enter) or clicks the OK button, the macro will continue on with the command following the PAUSE or PauseKey command.

Normally, when the user clicks the Cancel button, a cancel condition occurs in the macro, the macro is stopped, and the message "The macro is being canceled at user request." will appear. This Cancel condition can be captured with the ONCANCEL command. (See Lesson 21, *Handling Error, Cancel, Not Found, and Custom Conditions* for information on the ONCANCEL command.)

Because of some odd behavior in PROMPT dialog boxes, I recommend that you use `MessageBox` rather than `PROMPT` for most purposes.

## Examples and Exercises:

---

### Example 3-1 Combining text and a variable in a PROMPT display

```
vAddressNum:=16
PROMPT("RESULTS"; "You have added "+vAddressNum+" addresses to the database.")
PAUSE
```

---

Sometimes you may want your `PROMPT` message to display on more than one line. But if you include a hard return in your `PROMPT` command, a syntax error will occur when you compile the macro. ("A string constant cannot span a line boundary.") However, there is a sly, computer-nerdish way to force a carriage return in your message.

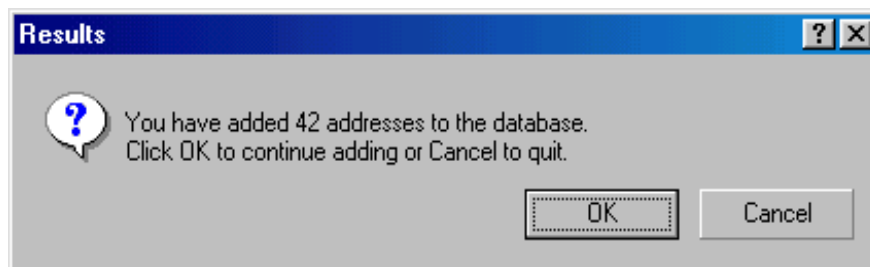
To do this, you must first create a variable that contains the NTOC value for a hard return (`NTOC(63754)`). You can use any valid name for the variable. I usually use `HRT` as the variable name. Next, you use that variable wherever you want a hard return in your `PROMPT` message.

The example below creates a variable called `HRT` that contains the value for a hard return, then uses it to cause a hard return in a `PROMPT` message. (Lesson 4, *Variables*, has more information on variables and Lesson 15, *Number to Character Values* has more information on the `NTOC` command.)

---

### Example 3-2 Adding a hard return to a PROMPT message

```
HRT:=NTOC(63754)
vAddressNum:=42
PROMPT("Results"; "You have added "+vAddressNum+" addresses to the database."+HRT+"Click
OK to continue adding or Cancel to quit."; QuestionMark!)
PauseKey(Enter!)
```



---

**Example 3-3 A PROMPT with an icon and left and top positioning**

---

```
PROMPT("Prompt Test"; "This is the message.";4; 0; 200)
PauseKey(Enter!)
```

---

**MESSAGEBOX**

**Syntax:** `vEnum:=MESSAGEBOX(<"Caption">; "Message"; <Style>; <{ParameterData}>) or MESSAGEBOX(<Status>; <"Caption">; "Message"; <Style>; <{ParameterData}>)`

**Purpose:** The MESSAGEBOX command is easily the most useful of all the commands in this lesson, and also the most versatile. I also believe it's the best one to use in most cases. This command allows you to create a dialog box that is very similar to the PROMPT command, but allows you to specify which buttons will appear on the dialog. As with the PROMPT command, you can also specify an icon that should appear, which in turn controls the sound that is played when the MessageBox is displayed.

**Return:** A value is returned in the variable. This may be the *Status* parameter, or to the specified variable, depending on the syntax used.

0	Error!	4	RetryButton!
1	OKButton!	5	IgnoreButton!
2	CancelButton!	6	YesButton!
3	AbortButton!	7	NoButton!

**Parameters:**

**Status:** The *Status* parameter is a variable name that you supply. Once the MESSAGEBOX is dismissed, this variable will contain a value based on the button the user clicked to dismiss the MESSAGEBOX dialog. The possible values are:

0	Error!	4	RetryButton!
1	OKButton!	5	IgnoreButton!
2	CancelButton!	6	YesButton!
3	AbortButton!	7	NoButton!

**Caption:** The *Title* parameter allows you to specify the text that will appear on the title bar of the dialog box. You should include the title in quotation marks unless you are specifying a variable that contains the title.

**Message:** The *Message* parameter will contain the message that will appear on the dialog box. As with the title, this message should be included in quotation marks. The message may also contain parameter indicators. A parameter indicator is a code that specifies additional items to be displayed in the message area. This is useful when you want text that is assigned to a variable to appear in the message.

The parameter indicator starts with a caret (^) and is followed by a number from 0 to 9. Up to 10 parameter indicators can be specified. You must also specify the *HasParameters!* style in the *Style* parameter. If you omit the *HasParameters!* style, the carets and numbers will appear in the MESSAGEBOX. You'll specify the actual text to display later in the command.

**Style:** The *Style* parameter allows you to specify various properties for your dialog box.

You can choose the buttons and icon to include on the message box, as well as the modality of your message box. The possible button styles are:

1	AbortRetryIgnore!	Abort, Retry, and Ignore buttons
2	OK!	OK button (default)
3	OKCancel!	OK and Cancel buttons
4	RetryCancel!	Retry and Cancel buttons
5	YesNo!	Yes and No buttons
6	YesNoCancel!	Yes, No, and Cancel buttons

You can place an icon on the message box. These icons are the same as those accessible in the PROMPT command, plus a couple of additional ones. The possible icon styles are:

0	IconNone!	No icon	(Default sound)
8	IconAsterisk!	Asterisk icon	(Asterisk sound)
16	IconExclamation!	Exclamation icon	(Exclamation sound)
16	IconWarning!	ExclamationIcon	(ExclamationSound)
24	IconHand!	Hand icon	(Critical Stop sound)
32	IconInformation!	Lowercase "I" icon	(Asterisk sound)
40	IconQuestion!	Question mark icon	(Question sound)
48	IconStop!	Stop sign icon	(Critical Stop sound)
48	IconError!	Stop sign icon	(Critical Stop sound)

When you use one of the above icons in a prompt, the sound used for that particular event (as shown above) will be played when the message box is displayed. For example, if in your Windows 95 Sound settings you have the DING.WAV file selected for the Exclamation event, the ding will sound when a prompt using the ExclamationPoint! Style is displayed.

To check your Windows system for selected sounds, click the Start button, then click **Settings > Control Panel**. Double-click the Sounds icon to list the sounds. The sounds assigned to the Asterisk, Critical Stop, Exclamation, and Question events will be played when the appropriate icon styles are used.

When any dialog box is displayed on the screen, one of the buttons has focus. If you do not specify which button should be the default, the first one will automatically have focus. You can specify which button will be the default button on MESSAGEBOX dialogs with the following styles:

64	DefButton1!	The first button is the default
128	DefButton2!	The second button is the default
192	DefButton3!	The third button is the default
256	ApplicationModal!	This is the default modality. You are allowed to switch to another application while the MESSAGEBOX is displayed, but you must dismiss the MESSAGEBOX before the macro will resume.
512	SystemModal!	You are not allowed to switch to another application, and you must dismiss the MESSAGEBOX before the macro resumes.
768	TaskModal!	This works the same as ApplicationModal! where you can switch to other applications, but you must dismiss the MESSAGEBOX dialog before the macro resumes.
1024	Beep!	You can also cause a beep when the dialog appears with the Beep! style:
2048	HasParameters!	When you specify parameter indicators ( the ^ and number) in the Message parameter, you must include the HasParameters! style.

You must then specify the desired text or variables in the *ParameterData* parameter.

4096 DontRetry  
8192 NoSafety  
16384 Recordable  
32768 NoFocus

**ParameterData:** The *ParameterData* parameter is used to specify text or variables that you want to appear in the message area of the MESSAGEBOX dialog. You can specify up to 10 items in this parameter. The first one is numbered 0; the second, 1; the third, 2; and so forth. These numbers correspond to the values used in the parameter indicators in the *Message* parameter. If you specify numbers in the parameter indicators that do not exist in the *ParameterData* parameter, the message “Out of Memory” will be displayed.

### Examples and Exercises:

In the example below, there are two parameter indicators in the *Message* parameter: ^0 and ^1. You can include up to 10 of these parameter indicators with numbers 0 through 9. Next, you will notice two items in the *Parameters* parameter. The first item corresponds to the parameter indicator with the lowest number, the second with the next lowest number, and so forth. The parameter indicators need not be in the same order as the parameters. They will match up automatically according to the parameter indicator numbers. They must, however, be consecutive. For instance, you cannot have a ^0 and a ^5, but none in between.

All of the optional parameters must be listed within French braces ( { } ) and separated with semicolons.

---

### Example 3-4 The MESSAGEBOX command using parameters

```
HRT:=NTOC(63754)
vText1:="This is the first message."
vText2:="This is the second message."
MESSAGEBOX(vStatus; "This is a MessageBox"; "Message: "+HRT+"^0 "+HRT+" ^1";
IconStop!+YesNo!+HasParameters!+DefButton2!; {vText2; vText1})
```



We’ve assigned some text to variables called vText1 and vText 2 in the above example. The next lesson will talk about variables in detail.

## MacroStatusPrompt

**Syntax:** `MacroStatusPrompt(State; <"Prompt">)`

**Purpose:** The MacroStatusPrompt command is a product command that will display your message on the status bar of a document screen. A MacroStatusPrompt message will overwrite all other status bar information, such as the font and page number. Unlike a PROMPT command, if you want the user to type into a document while the MacroStatusPrompt is displayed, they need not click in the document first before typing information. A MacroStatusPrompt message will remain on the status bar until another MacroStatusPrompt command removes or overwrites it, or until the macro is completed.

**Return:** None

### Parameters:

**State.** The *State* parameter is required. It tells WordPerfect whether to display or erase the message on the status bar. There are two possible states. Use **Off!** when you want to remove a MacroStatusPrompt message from the screen, and **On!** when you are posting a new message.

- 0 Off! Turn off the message
- 1 On! Display the message

**Prompt.** The *Prompt* parameter is optional. It contains the actual message you want to appear on the status bar. The message can only be as long as the status bar, so the message length is limited by the screen resolution. Keep this in mind if you are writing macros that may have to run on several different computers. When you want to remove a MacroStatusPrompt from the screen, leave this parameter blank.

### Remarks:

The MacroStatusPrompt command is a WordPerfect product command.

### Examples and Exercises:

---

#### Example 3-5 The MacroStatusPrompt command

```
MacroStatusPrompt(On!; "This line will appear on the status line of your screen.")
PauseKey(Enter!)
MacroStatusPrompt(Off!)
```

---

## PauseKey

**Syntax:** `PauseKey(Key; <Character>)`

**Purpose:** PauseKey causes your macro to pause. It can be used just after a PROMPT or MacroStatusPrompt command to pause the macro, or anywhere else you would like the macro to pause.

**Return:** None

### Parameters:

**Key.** The *Key* parameter tells WordPerfect which key the user will press to end the pause. Generally, this key will be the (Enter) key, but in some cases you may prefer to use Close, Escape, or a character key.

The *Key* parameter is required. It can contain one of the following choices:


- |   |            |  |
|---|------------|--|
| 0 | Enter!     | Pressing (Enter) terminates the pause.   |
| 1 | Cancel!    | Pressing (Escape) terminates the pause.  |
| 2 | Close!     | Choosing <u>C</u> lose from the <u>F</u> ile menu ends the pause.  |
| 3 | Character! | Typing the character specified in the Character parameter after clicking in the document ends the pause. |
| 4 | Any!       | Pressing any character key after clicking in the document ends the pause.                                |

**Character.** The *Character* parameter is optional and is only used when the *Key* parameter is set to **Character!**. Specify the character you want the user to press to end the pause. It is most common to use a character to terminate a pause when users are entering lots of text during the pause. They can then hit a specified key to continue without moving their hands off the keyboard.

### Remarks:

For example, to end the pause with **Cancel** key (Escape), use the command **PauseKey(Cancel!)**. This specifies that the terminating key should be **Escape**. The user can end the pause by pressing Escape.

**PauseKey(Any!)** can be used to pause the macro, wait for any keystroke and then continue. The key pressed will be assigned to an internal system variable called **?PauseKey**. The value returned will be numeric. See Appendix I, *PauseSet and PauseCommand Tokens* for a list of possible values. You can then test the user's input with this system variable.

**PauseKey** can be inserted into a macro while you are recording a macro. To do this, click **T**ools > **M**acro > **P**ause from the menu, or click the Pause button  on the Macro Feature Bar. A checkmark will appear next to **Pause** on the menu, and the message **Record Pause** will appear on the status bar. When you see this message, WordPerfect allowing you to perform keystrokes or commands, but is not recording them. To end the pause, click **T**ools > **M**acro > **P**ause (or the Pause button ) again.

When you are paused, notice that the **C**ommands button on the Macro Feature Bar becomes enabled, and is disabled when you are not paused. Any keystrokes or commands you perform while the macro is in a paused state will not be recorded in your macro. After you turn Pause off, recording will resume.

When you play a macro with a **PauseKey** command in it and you get to the pause, the message **Play Pause** will appear on the status bar. When you see this message, WordPerfect is waiting for you to perform some action, then press (Enter) or click **T**ools > **M**acro > **P**ause on the menu.

### Examples and Exercises:

---

#### Example 3-6 Examples of the **PauseKey** command

```
PauseKey(Enter!)
PauseKey(Cancel!)
PauseKey(Close!)
PauseKey(Character!; "/")
PauseKey(Any!)
```

---

#### Exercise 3-1 A Memo Macro

Imagine you are making a macro that will create an office memorandum. You'll start by recording keystrokes for the main portion of the macro. Then you'll edit the macro to jazz it up with **PROMPT** commands. From a blank document, press **Record Macro** (Ctrl+F10) or click **T**ools > **M**acro > **R**ecord to start a new macro. Type *mymemo*, and press (Enter). Following the keystrokes below, create the *mymemo* macro:

- 1 Click **F**ormat > **L**ine > **T**ab Set.
- 2 Click **C**lear All.
- 3 In the Position edit window, type 0.75", click **S**et, then OK.
- 4 Click **F**ormat > **L**ine > **C**enter.
- 5 Type **M E M O R A N D U M**, and (Enter) twice.
- 6 Type **TO:**, (Tab). Click the Pause button on the Macro Feature Bar twice.
- 7 Press (Enter) twice.
- 8 Type **FROM:**, (Tab). Click the Pause button twice.
- 9 Press (Enter) twice.
- 10 Type **DATE:**, (Tab), then press Ctrl+D to insert date text.
- 11 Press (Enter) twice.
- 12 Type **RE:**, (Tab). Click the Pause button twice.
- 13 Press (Enter) twice.
- 14 Click **F**ormat > **L**ine > **C**enter.
- 15 Click **E**dit > Repeat **n**ext action. Type 45 and (Enter).
- 16 Press \*.
- 17 Press (Enter) twice.

Now press **Record Macro** (Ctrl+F10) to end recording of your keystrokes. Notice that even though you ended the macro, the information you typed during **Record Macro** is still on the screen.

Click in the document, then click **F**ile > **C**lose to clear the screen,. You need not save the document. Now, play the macro. From a blank document, press **Play Macro** (Alt+F10) or click **T**ools > **M**acro > **P**lay, type *mymemo*, and press (Enter). After the macro types TO:, enter some text and press (Enter). The macro will now type FROM:. Type some text and (Enter). Continue until the asterisk line is typed on the screen and the macro ends.

Now you'll edit this macro to add some PROMPT messages. Click **T**ools > **M**acro > **E**dit. Type *mymemo*, and press (Enter). You should now have the macro on the screen. Your commands may not appear in exactly the same places as shown below, but as long as all the commands are in their proper order, your macro is correct.

Place the insertion point after the first Tab command. Press (Enter) to add a blank line. Type **PROMPT** ("RECIPIENT'S NAME"; "Enter RECIPIENT'S name: "). Add the other PROMPT commands as shown in bold below. Click the Save & **C**ompile button when done to save and recompile the macro.

```
Application (WordPerfect; "WordPerfect"; Default!; "EN")
TabSet(Origin: Relative!; Position: 1.75"; TabType: TabLeft!)
Center()
Type(Text: "M E M O R A N D U M")
HardReturn()
HardReturn()
Type(Text: "TO:")
Tab()
PROMPT("RECIPIENT NAME"; "Enter RECIPIENT'S name (Press (Enter)).")
PauseKey(Key: Enter!)
HardReturn()
HardReturn()
Type(Text: "FROM:")
Tab()
PROMPT("SENDER NAME"; "Enter SENDER'S name (Press (Enter)).")
PauseKey(Key: Enter!)
```

```

HardReturn()
HardReturn()
Type(Text: "DATE:")
Tab()
DateText()
HardReturn()
HardReturn()
Type(Text: "RE:")
Tab()
PROMPT("SUBJECT"; "Enter SUBJECT (Press (Enter)).")
PauseKey(Key: Enter!)
HardReturn()
HardReturn()
Center()
RepeatValue(Value: 45)
Type(Text: "**")
HardReturn()
HardReturn()

```

Now, when you run the *mymemo* macro, it will prompt you for the information to be filled in on the memo. After the first PROMPT message appears, click in the document next to TO: and type a name. **Note: A bug in WordPerfect makes the PROMPT disappear behind the WordPerfect window.**

Press (Enter) when you have finished typing. The FROM: will be typed in for you, and the PROMPT dialog should reappear with a prompt to type the sender's name. Type a name and press (Enter). Continue until the macro is completed.

Now edit the macro again and try adding *Left* and *Top* values, such as 150 and 30 to position the PROMPT dialog in a less intrusive area of your screen. Run the macro to see where the box will appear.

## PAUSE

**Syntax:** PAUSE

**Purpose:** The PAUSE command works just like a `PauseKey(Enter!)` command. Pressing (Enter) normally ends the PAUSE, and the next command after the PAUSE is invoked. The difference between the two is that while `PauseKey(Enter!)` is a product command that is built into WordPerfect, PAUSE is a programming command. Thus, you can use PAUSE even in macros that do not interact with WordPerfect.

**Return:** None

**Parameters:** None

**Remarks:**

The PAUSE command has no parameters to set an ending keystroke like the `PauseKey` command does, but you can set different options for the PAUSE command with the `PauseSet` command.

The PAUSE command is not recordable; you cannot add it while you are recording a macro. You must edit the macro and type the command where you'd like the PAUSE to occur. However, you can end a PAUSE when you're playing a macro by clicking **Tools > Macro > Pause**, or by pressing (Enter) just as you can with a `PauseKey(Enter!)` command.

Sometimes PerfectScript forgets which key is supposed to end a PAUSE. If you find the expected key isn't working, you can click the **OK** or **Cancel** button on the PROMPT dialog, or click **Tools > Macro > Pause**.

## PauseSet

**Syntax:** `PauseSet (Token)`

**Purpose:** The PauseSet command allows you to set a different key to end a PAUSE command. The default key to end a PAUSE is (Enter).

**Return:** None

### Parameters:

**Token:** You must specify the token that you want to end the PAUSE in the *Token* parameter. Any macro token is valid for a PauseSet. However, you should only use tokens that are easily accessible with one keystroke by the user. Some possible examples are FontDlg (F9), PosScreenUp (PgUp), Tab (Tab), PosPagePrevious (Alt+PgUp), or HardReturn (Enter). Character keys may not be used.

### Remarks:

You can set the *Token* that will end a PAUSE with a PauseSet command, and then every PAUSE command will end when that token is executed for the rest of the macro, or until another PauseSet command changes the *Token*. A list of keystrokes and the command tokens they generate can be found in Appendix I, *PauseSet and PauseCommand Tokens*.

A PauseSet and PAUSE command used together work similarly to the PauseCommand command.

*Note: For the PauseSet token to work, you need to click in the document first before pressing the specified key if it was used with a PROMPT. That pesky bug that makes the PROMPT go behind WordPerfect is still at work here, so doing so makes the PROMPT disappear from view.*

### Examples and Exercises:

---

#### Example 3-7 The PauseSet command

```
HRT:=NTOC(63754)
PauseSet(Tab)
PROMPT("Example"; "This is an example of how using the PauseSet command is supposed to
work."+HRT+" Click in the document, then press (Tab) to continue.")
PAUSE
Type("The macro is done.")
```

---

## PauseCommand

**Syntax:** `PauseCommand (Token)`

**Purpose:** The PauseCommand command works just like the combination of a PauseSet and PAUSE command. The token you specify works just for that instance of the command. Unlike PauseSet, the token you specify does not carry forward to the next instance of PAUSE or PauseCommand.

**Return:** None

### Parameters:

**Token:** You must specify the token that you want to end the pause in the PauseCommand's *Token* parameter. Any macro token is valid for a PauseCommand. However, you should only use tokens

that are easily accessible with one keystroke by the user. Some possible examples are `FontDlg` (F9), `PosScreenUp` (PgUp), `Tab` (Tab), `PosPagePrevious` (Alt+PgUp), or `HardReturn` (Enter). Character keys may not be used. A list of keystrokes and the command tokens they generate can be found in Appendix I, *PauseSet and PauseCommand Tokens*.

## Examples and Exercises:

---

### Example 3-8 The `PauseCommand` command

```
HRT:=NTOC(63754)
PROMPT("Example"; "This is an example of the PauseCommand command." + HRT + "Click in the
document then press (Alt+PgUp) to continue.")
PauseCommand(PosPagePrevious)
Type("The macro is done.")
```

---

*Note: For the `PauseCommand` token to work, you need to click in the document first before pressing the specified key if it was used with a `PROMPT`. That pesky bug that makes the `PROMPT` go behind WordPerfect is still at work here, so doing so makes the `PROMPT` disappear from view.*

## MacroPause

**Syntax:** `MacroPause ()`

**Purpose:** The `MacroPause` command works just like the `PAUSE` and `PauseKey(Enter!)` commands. Pressing (Enter) or clicking `Tools > Macro > Pause` will end the pause condition.

**Return:** None

**Parameters:** None

## Examples and Exercises:

---

### Example 3-9 The `MacroPause` command

```
HRT:=NTOC(63754)
PROMPT("Example"; "This is an example of the MacroPause command." + HRT + "Click in the
document then press (Enter) to continue.")
MacroPause()
Type("The macro is done.")
```

---

### Exercise 3-2 Using `MacroStatusPrompt` and `PauseCommand`

Edit your `mymemo` macro by clicking `Tools > Macro > Edit`. Type `mymemo`, and (Enter). Change all the `PROMPT` commands to `MacroStatusPrompt` commands. Change the `PauseKey` commands to `PauseCommand` commands. Choose any token you like, such as `Tab`, `HardReturn`, or `FontDlg`.

```
Application (WordPerfect; "WordPerfect"; Default!; "EN")
TabSet(Relative!;{1.75";TabLeft!}) Center()
Type("M E M O R A N D U M")
```

```

HardReturn() HardReturn()
Type(Text: "TO:")
Tab()
MacroStatusPrompt(On!;"Enter RECIPIENT'S name (Press Tab).")
PauseCommand(Tab)
HardReturn()
HardReturn()
Type(Text: "FROM:")
Tab()
MacroStatusPrompt(On!; "Enter SENDER'S name (Press Tab).")
PauseCommand(Tab)
HardReturn()
HardReturn()
Type(Text: "DATE:")
Tab()
DateText()
HardReturn() HardReturn()
Type(Text: "RE:")
Tab()
MacroStatusPrompt(On!; "Enter SUBJECT (Press Tab).")
PauseCommand(Tab)
HardReturn() HardReturn()
Center()
RepeatValue(45)
Type(Text: "**")
HardReturn() HardReturn()

```

When you have finished editing the macro, click the Save & Compile button. When the Play button is enabled, your macro is ready to play. Press **Ctrl+N** to go to a new screen. Press **Macro Play** (Alt+F10) or click Tools > Macro > Play, type *mymemo*, and (Enter). Notice the messages that appear on the status bar.

You will probably find that the `MacroStatusPrompt` is less awkward than `PROMPT` for this type of macro, but the message it creates is also less noticeable to the user. In later lessons, we'll talk about other commands that make handling user input easier for the macro programmer, and friendlier to the user.

## WAIT

**Syntax:** `WAIT (TenthsOfSecond)`

**Purpose:** The `WAIT` command can be used to make a macro stop for a specified amount of time at any point in a macro. For instance, if you would like a `MacroStatusPrompt` or `PROMPT` message to be displayed for a few seconds so the user can read it, but then have the macro continue without user intervention, insert a `WAIT` command just after the message command.

**Return:** None

### Parameters:

***TenthsOfSecond*** The length of time to wait is specified in tenths of a second. If you want the macro to wait ten seconds, for instance, use the command `WAIT(100)`. For five seconds, use `WAIT(50)`. The maximum time that one `WAIT` command can wait is 60 seconds; thus `WAIT(600)` will be the longest

interval for one WAIT command. If more than 60 seconds is needed, you can use multiple WAIT commands. If you specify a number larger than 600, an error will occur when you play the macro.

### Examples and Exercises:

---

#### Example 3-10 The WAIT command

```
MacroStatusPrompt(On!; "This message will display for 20 seconds.")
WAIT(200)
MacroStatusPrompt(Off!)
```

---

## BEEP

**Syntax:** BEEP (<BeepType> )

**Purpose:** Users are more likely to pay attention to a macro's progress if they receive both a visual and an audible cue for action. The PROMPT command displays a visual message or request for information. The BEEP command generates an audible beep from the computer. This, along with a message displayed on the screen, is sure to get the user's attention. Place the BEEP command just before your PROMPT or other command where you would like a beep to sound.

*You may have changed the beep sound in Windows. If you have Microsoft Plus, it is very likely your sounds have been changed from the default sounds. If this is the case, you may get a sound other than a beep. To see your selected sounds, click the **Start** button, then **Settings > Control Panel**. Select **Sounds** to see the list of sounds.*

**Return:** None

**Parameters:**

**BeepType:** Specify the type of sound to make. The possible choices are:

- |   |              |  |
|---|--------------|--|
| 0 | None         | no beep will sound   |
| 1 | Standard!    | a standard beep from the built-in speaker                          |
| 2 | Default!     | the sound selected for Default Sound in Sound Properties (default) |
| 3 | Question!    | the sound selected for Question in Sound Properties                |
| 4 | Asterisk!    | the sound selected for Asterisk in Sound Properties                |
| 5 | Exclamation! | the sound selected for Exclamation in Sound Properties             |

---

#### Exercise 3-3 Adding BEEP to the Memo Macro

Edit the *mymemo.wcm* macro again. Add the BEEP command just before each MacroStatusPrompt command, as shown in the listing below.

```
Application(WordPerfect; "WordPerfect"; Default!; "EN")
TabSet(Relative!;{1.75";TabLeft!})
Center()
Type("M E M O R A N D U M")
HardReturn
HardReturn
Type(Text: "TO:")
Tab()
BEEP(Default!)
MacroStatusPrompt(On!;"Enter RECIPIENT'S name (Press Tab).")
```

```

PauseCommand(Tab)
HardReturn
HardReturn
Type(Text: "FROM:")
Tab()
BEEP(Question!)
MacroStatusPrompt(On!;"Enter SENDER'S name (Press (Tab).)")
PauseCommand(Tab)
HardReturn
HardReturn
Type(Text: "DATE:")
Tab()
DateText()
HardReturn
HardReturn
Type(Text: "RE:")
Tab()
BEEP(Asterisk!)
MacroStatusPrompt(On!;"Enter SUBJECT (Press Tab).)")
PauseCommand(Tab)
HardReturn
HardReturn
Center()
RepeatValue(45)
Type(Text: "**")
HardReturn
HardReturn

```

Play the macro from a blank document screen. You will now hear BEEPS, as well as see the message prompting you to enter information.

---

Use BEEP with care. If your macro BEEPs too much it can be very annoying to the user, and it can also be very disruptive to people sitting around the user.

## ENDPROMPT

**Syntax:** ENDPROMPT

**Purpose:** The ENDPROMPT command is used to terminate a PROMPT message. Normally, this command is not needed since the PROMPT message will disappear when the PAUSE or PauseKey command that follows the PROMPT is ended, or when the WAIT command is completed. But in rare circumstances when the PROMPT may not disappear by itself, this command is available to erase the message for you.

**Return:** None

**Parameters:** None

### Examples and Exercises:

---

#### Example 3-11 The ENDPROMPT command

```
PROMPT("Prompt Test"; "This is the message."; 4)  
WAIT(40)  
ENDPROMPT
```

---

---

## Quiz

1. What command can be used to audibly get the user's attention?  
\_\_\_\_\_
2. What command will post a message on the status bar of the document screen that will stay until you remove it?  
\_\_\_\_\_
3. What two commands used together will display a message for 3 seconds?  
\_\_\_\_\_
4. What three commands used together will display a message until the user presses (F9)?  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_
5. How can you get the same results (as in #4) with just 2 commands?  
\_\_\_\_\_  
\_\_\_\_\_
6. How can you insert a **PauseKey** command while *recording* a macro?  
\_\_\_\_\_
7. What command(s) can you specify if you want the user to press (Tab) to end a **PAUSE**?  
\_\_\_\_\_  
\_\_\_\_\_
8. What command can you use to display a dialog containing a message with a Yes and No button and a question mark icon?  
\_\_\_\_\_