

Mouse commands

Mouse movement is very important for many Windows applications. Mouse movement has many variations and isn't always handled gracefully with the built-in commands of NaturallySpeaking.

There are also mice such as the "Head Mouse" and variants thereof which can be very helpful but which need different types of mouse commands to be most useful when used with NaturallySpeaking. Generally-speaking, with these mice verbal commands are useful for clicking and holding the mouse buttons, but mouse movement is done with the mouse itself.

These are illustrated with sample commands with limited commentary. Other special situations occur, this is a collection of samples of such mouse movement.

Holding Buttons

Particularly using a head mouse or for someone able to use a trackball but unable to hold a button while using these devices, commands to hold down and release buttons can be helpful.

These "hold" macros keep the mouse button depressed until the "release" command is given. Variants of these commands were instrumental in allowing the cover of "Scripting for Dragon NaturallySpeaking 9" to be drawn along with a head mouse.

Command: <releasehold><leftrightcenter>

Hold Left	Release Left	
Hold Right	Release Right	
Hold Center	Release Center	Center and Middle behave the same
Hold Middle	Release Middle	

```
Private Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, _  
ByVal dx As Long, ByVal dy As Long, ByVal cButtons As Long, _  
ByVal dwExtraInfo As Long)
```

```
Type POINTAPI 'Declare types  
x As Long  
y As Long  
End Type  
,
```

```
Declare Function GetCursorPos Lib "user32" _
```

(lpPoint As POINTAPI) As Long 'Declare API

Const MOUSEEVENTF_LEFTDOWN = &H2

Const MOUSEEVENTF_LEFTUP = &H4

Const MOUSEEVENTF_MIDDLEDOWN = &H20

Const MOUSEEVENTF_MIDDLEUP = &H40

Const MOUSEEVENTF_RIGHTDOWN = &H8

Const MOUSEEVENTF_RIGHTUP = &H10

Sub Main

Dim z As POINTAPI 'Declare variable

Dim xpos As Long, ypos As Long

GetCursorPos z 'Get Coordinates

xpos = CLng(z.x)

ypos = CLng(z.y)

If ListVar1 = "Release" Then

 Select Case ListVar2

 Case "Left"

 mouse_event MOUSEEVENTF_LEFTUP, xpos, ypos, cButt, dwEI

 Case "Right"

 mouse_event MOUSEEVENTF_RIGHTUP, xpos, ypos, cButt, dwEI

 Case "Center", "Middle"

 mouse_event MOUSEEVENTF_MIDDLEUP, xpos, ypos, cButt, dwEI

 End Select

Else

 Select Case ListVar2

 Case "Left"

 mouse_event MOUSEEVENTF_LEFTDOWN, xpos, ypos, cButt, dwEI

 Case "Right"

 mouse_event MOUSEEVENTF_RIGHTDOWN, xpos, ypos, cButt, dwEI

 Case "Center", "Middle"

 mouse_event MOUSEEVENTF_MIDDLEDOWN, xpos, ypos, cButt, dwEI

 End Select

End If

End Sub

Mouse + Keystroke

Some applications expect one to be able to click a mouse and a modifier key (Ctrl, Shift, or Alt) at the same time.

A complication to some of these commands is that Dragon NaturallySpeaking by default uses the Ctrl by itself to indicate that only commands should be recognized, and the Shift by itself to indicate that only dictation should be recognized. If these commands are to be used, it is suggested that those default options be modified to not have any keystroke force command or dictation recognition.

These commands are difficult to test without applications that actually use these combinations, please advise if they are not operating successfully.

Command: Shift Click Left
(Variants of this to come later)

Option Explicit

```
Type POINTAPI 'Declare types
x As Long
y As Long
End Type
```

```
Declare Function GetCursorPos Lib "user32" _
(lpPoint As POINTAPI) As Long 'Declare API
```

```
Dim z As POINTAPI 'Declare variable
```

```
Sub Main
GetCursorPos z 'Get Coordinates
SendKeys "+{ClickLeft " & z.x &"," & z.y &"}",1
End Sub
```

Holding Shift/Ctrl Key

In conjunction with mouse movement, you may wish to hold the Shift Key or the Control Key down. This only works if you have changed the Dragon HotKeys for recognizing dictation or commands to something other than the “Shift” key or the “Ctrl” key respectively.

Command Name: "<shiftctrl> Key <dirupdown>"

Option Explicit

Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte, ByVal bScan As Byte, ByVal dwFlags As Long, ByVal dwExtraInfo As Long)

Const KEYEVENTF_KEYUP = &H2

Const VK_SHIFT = &H10

Const VK_CONTROL = &H11

Sub Main()

' Press or release the button.

,

Select Case ListVar1

Case shift

Modifier = VK_SHIFT

Case control

Modifier = VK_CONTROL

End Select

Select Case ListVar2

Case down

keybd_event Modifier, 0, 0, 0 '

Case up

keybd_event Modifier, 0, KEYEVENTF_KEYUP, 0

End Select

DoEvents

End Sub

Mouse Action With SendKeys

Command name: <dirleft> Button <dirupdown>
(Say "Left Button Down", "Right Button Up" etc)

Option Explicit

Type POINTAPI 'Declare types

x As Long

y As Long

End Type

Declare Function GetCursorPos Lib "user32" _
(lpPoint As POINTAPI) As Long 'Declare API

Dim z As POINTAPI 'Declare variable

Sub Main

```

GetCursorPos z 'Get Coordinates
'SendKeys DownLeft, DownRight etc.:
SendKeys "{" & ListVar2 & ListVar1 & " " & z.x & "," & z.y & "}",1
End Sub

```

Multiple Window/Screen Control

Sometimes one needs to go back and forth between two windows. This command is intended to help one go back to the same spot after doing something in another window. By itself it is not useful. Add code to act on the 2nd window and it may be extremely useful. Originally a Page Down was done in the 2nd window. In another case a single field was incremented by one, processed, and then control is returned to the original window.

Command Name: sample two screen action

Description: This macro preserves the current cursor and mouse positions, then can do some other arbitrary action. It was originally written to switch screens, do a Page Down on another application, then return to the original position.

Option Explicit

```

Type POINTAPI
x As Long
y As Long
End Type

```

```

' declare Win32 API functions

```

```

Declare Function GetCursorPos Lib "user32" _
  Alias "GetCursorPos" (lpPoint As POINTAPI) As Long

```

```

Declare Function SetCursorPos Lib "user32" _
  Alias "SetCursorPos" (ByVal x As Long, ByVal y As Long) As Long

```

```

Declare Function GetForegroundWindow Lib "user32" ()

```

```

Declare Function SetForegroundWindow Lib "user32" (ByVal hwnd As Long) As Long

```

```

Sub Main
Dim pt As POINTAPI
Dim hwnd As Long

```

```

hwnd = GetForegroundWindow()

```

```

GetCursorPos pt          ' Get the current cursor position

```

```
'  
' At this point, do whatever other action is needed, including moving elsewhere  
'   Finish doing arbitrary action  
  
Wait .2  
  
SetCursorPos(pt.x, pt.y)      ' return the cursor position to the original position  
Wait .2  
SetForegroundWindow(hwnd)    ' to return focus to original application  
  
End Sub
```