MouseOverControls

Table of contents

Mouse Over Controls

How To

Introduction

Getting Started Top Next

Mouse over controls (Version 2).

This help file will show you how to check -

1. If the mouse is over a window.
2. If the mouse is over a specified control.
3. If the mouse is over any control
4. If the mouse is over any specified control type
5. If the window has been moved or re-sized.
6. If the window was minimised.
7. If the window was maximised.
8, If the window is neither minimised or maximised.
9. If the window exists.

and...

10. Delete a control.

It will detect automatically if the window has been moved or re-sized and give the controls their new co-ordinates in that window.

Installation

Installation Previous Top Next

Install Location.

The **MouseOverControls.inc** file must be placed in either -

1. The same directory as the application source file

or

2. The IWBasic/EBasic /inc directory.

Commands

These are the commands available.

1. MouseOverControl (WINDOW MyWindow, INT MyControl, STRING MyWinDesc, STRING MyControlType)

The above command is used for storing the coordinates and window details of a control we want to keep track of. (Used later to check if the mouse is over the stated control).

Location - After a control has been created.

Example 1:

```
CONTROL win,@BUTTON,"Button 1",10,200,150,25,0,button_1
MouseOverControl (win,button_1,"win","B")
```

Where -

win - is the window
Button_1 - the control I want to track
"win" - a description (name) of the window the control belongs to.
"B" - denotes a button control.

Example 2:

```
CONTROL win,@BUTTON,"Button 1",10,200,150,25,0,button_1
MouseOverControl (win,1,"win","B")
```

Where -

win - is the window
1 - the control I want to track
"win" - a description (name) of the window the control belongs to.
"B" - denotes a button control.

Note:- Each control needs a description for the window it belongs to, this description has to be the same as the one used in the TrackWindow command plus the control type.

The control type parameter can be any one of the following:

B - Button
C - Checkbox
E - Edit
L - Listbox
R - Radio
S - Static
V - Listview

Note:- You can use lower case letters if you wish.

Use the same window description for all the controls in that window, example, the window is called win, so we need to include "win". If our window was called win2, we include "win2" etc.

```
MouseOverControl(win,button_1,"win","B")
MouseOverControl(win,button_3,"win","B")
MouseOverControl(win,button_5,"win","")
MouseOverControl(win,radio_1,"win","R")
MouseOverControl(win,check_1,"win","C")
MouseOverControl(win,static_1,"win","S")
```

2. TrackWindow (WINDOW MyWindow, STRING MyWinDesc), INT

The above command tells you if the mouse is over the window but it also checks to see if the window has moved, if so it updates the controls with their new coordinates.

You can just use - TrackWindow(win,"win") on it's own if you do not
want to do anything whilst the mouse is over the window.

Returns 0 - Mouse not over the window, 1 - Mouse is over the window.

Location - MUST be in the CASE @IDTIMER section of your code.

Example 1:

```
CASE @IDTIMER

    IF TrackWindow(win,"win")

        SETCAPTION win,"Mouse is Over window"

    ELSE

        SETCAPTION win,"Mouse is Not Over window"

    ENDIF
```

Example 2:

```
CASE @IDTIMER
```

```
TrackWindow(win,"win")
```

Note:- You must use this command! otherwise if the window is moved or re-sized the coordinates of the controls will not be updated and the tracking will not work correctly.

You must use the same window description as used in the MouseOverControl commands for that window.

Example: MouseOverControls described as-

```
MouseOverControl(win,button_1,"win","B")
MouseOverControl(win,button_3,"win","B")
MouseOverControl(win,button_5,"win","")
MouseOverControl(win,radio_1,"win","R")
MouseOverControl(win,check_1,"win","C")
MouseOverControl(win,static_1,"win","S")
```

So TrackWindow has to be described as-

```
TrackWindow(win,"win")
```

Where "win" is the same description for both commands for that window.

3. MouseOver (WINDOW MyWindow, INT MyControl, STRING MyWindowDesc), INT

Tells you if the mouse is over the stated control.

Returns 0 - Mouse not over, 1 - Mouse moved over, 2 - Mouse remains over.

Location - MUST be in the CASE @IDTIMER section of your code.

Example 1:

```
IF MouseOver(win,static_1,"win") = 1 Do the following only once.

    SETSIZE win,10,140,700,35,static_1
    SETFONT win,"Tahoma", 22, 400, 0, static_1

ELSE

     MouseMovedOff ..........

ENDIF
```

Example 2:

```
IF MouseOver(win,static_1,"win") > 0 Keep doing the following.

    Keep doing this.....

ELSE
```

```
        MouseMovedOff ..........

    ENDIF
```

Note:- Must be used in conjunction with the MouseMovedOff command.

4. MouseMovedOff (WINDOW MyWindow, INT MyControl, STRING MyWindowDesc), INT

Tells you if the mouse has now moved off the stated control it was over.

Returns 0 - Mouse has not moved off the control, 1 - Mouse has moved off the control.

Location - MUST be in the CASE @IDTIMER section of your code.

Example:

```
    IF MouseOver(win,static_1,"win") = 1 Do the following only once.

        SETSIZE win,10,140,700,35,static_1
        SETFONT win,"Tahoma", 22, 400, 0, static_1

    ELSE

        IF MouseMovedOff(win,static_1,"win") Do the following only once.

            SETSIZE win,10,140,300,35,static_1
            SETFONT win,"Arial", 10, 400, 0, static_1

        ENDIF

    ENDIF
```

Note:- Must be used in conjunction with the MouseOver command.

The above example shows you how to correctly use the MouseOver and MouseMovedOff commands together.

5. MouseOverAnyControl(WINDOW ControlsWindowIn,STRING ControlsWindowDesc),INT

Returns 0 - Mouse is not over any controls, 1 - Mouse is over a control.

Location - MUST be in the CASE @IDTIMER section of your code.

Example:

```
    IF MouseOverAnyControl(win,"win") = 1

        SETCAPTION win,"Mouse is over a control"

    ENDIF
```

Tells you if the mouse is over any control in the specified window.

6. MouseOverControlType(WINDOW ControlsWindowIn,STRING ControlsWindowDesc,STRING ControlsType),INT

Returns 0 - Mouse is not over any controls of the specified type, 1 - Mouse is over a control of the specified type.

Location - MUST be in the CASE @IDTIMER section of your code.

The control type parameter can be any one of the following:

B - Button
C - Checkbox
E - Edit
L - Listbox
R - Radio
S - Static
V - Listview

Note:- You can use lower case letters if you wish.

Example:

```
IF MouseOverControlType(win,"win","B") = 1

   SETCAPTION win,"Mouse is over a button control"

ENDIF

IF MouseOverControlType(win,"win","S") = 1

   SETCAPTION win,"Mouse is over a static control"

ENDIF
```

Tells you if the mouse is over any control of the type specified in the window.

7. WindowMoved(), INT

Tells you if the window (which has the mouse over) has moved or been re-sized.

Returns 0 - Window has not been moved or re-sized, 1 - Window has been moved or re-sized.

Location - MUST be in the CASE @IDTIMER section of your code.

Example:

```
IF WindowMoved() = 1
```

```
        Do something...
    ELSE
        Do this...
    ENDIF
```

6. WindowHasFocus (WINDOW MyWindow), INT

Tells you if the window specified has focus (topmost or is being used).

Returns 0 - Not topmpost, 1 - Topmost.

Location - anywhere you want, but to continually check you should place it in a timer or maybe a loop.

Example:

```
    IF WindowHasFocus(win) = 1
        Do something...
    ELSE
        Do this...
    ENDIF
```

7. WindowExists (WINDOW MyWindow), INT

Tells you if a window exists or not.

Returns 0 - Window does not exist, 1 - Window exists.

Location - anywhere you want, but to continually check you should place it in a timer or maybe a loop.

Example:

```
    IF WindowExists(win) = 1
        Do something...
    ELSE
        Do this...
    ENDIF
```

8. WindowMinimised (WINDOW MyWindow, STRING MyWindowDesc), INT

or WindowMinimized (WINDOW MyWindow, STRING MyWindowDesc), INT

Tells you if a window in minimised or not.

Returns -1 (minus 1) - Window does not exist yet, 0 - Not minimised, 1 - Has just been minimised, 2- Remians minimised.

Location - anywhere you want, but to continually check you should place it in a timer or maybe a loop.

Example:

```
IF WindowMinimised(win,"win") = 1
    Do something...
ELSE
    Do this...
ENDIF
```

9. WindowMaximised (WINDOW MyWindow, STRING MyWindowDesc), INT

or WindowMaximized (WINDOW MyWindow, STRING MyWindowDesc), INT

Tells you if a window has been maximised.

Returns -1 (minus 1) - Window does not exist yet, 0 - Not maximised, 1 - Has just been maximised, 2 - Remains maximised.

Location - anywhere you want, but to continually check you should place it in a timer or maybe a loop.

Example:

```
IF WindowMaximised(win,"win") = 1
    Do something...
ELSE
    Do this...
ENDIF
```

10. WindowNormal (WINDOW MyWindow, STRING MyWindowDesc), INT

Returns -1 (minus 1) - Window does not exist yet, 0 - Minimised or maximised, 1 - Not minimised or maximised (normal window state).

Location - anywhere you want, but to continually check you should place it in a timer or maybe a loop.

Example:

```
IF WindowNormal(win,"win") = 1
    Do something...
ELSE
    Do this...
ENDIF
```

11. DeleteControl (WINDOW MyWindow, INT MyControl), INT

Deletes a control from a window.

Returns 0 - Not deleted, 1 - Deleted.

Location - anywhere.

Example:

```
DeleteControl(win,Button_1)
```

12. StopTracking()

Deletes any pointers used and clears memory.

Returns 0.

Location - MUST be before you exit your program and should always be used.

Example 1:

```
WAITUNTIL EndProgram = 1
GOSUB StopTracking
END
```

Example 2:

```
        CASE button_exit 'Exit

                IF @NOTIFYCODE = 0

                    StopTracking()

                    END

                ENDIF
```

Example

Example Previous Top Next

Example of how to use the commands.

Copy and paste from the example program not from here.

You need to include the MouseOverControls2.inc file in your code for the mouse over controls function to work.

You MUST place this include file AFTER Windowssdk.inc if Windowssdk.inc is used.

```
$include "windowssdk.inc"
$include "MouseOverControls2.inc"


OPENWINDOW win,100,100,900,500,@CAPTION|@MINBOX|@MAXBOX,0," Mouse over controls - screen 1.",&Handler
```

Create your controls as normal...

```
CONTROL win,@BUTTON,"Button 1",10,200,150,25,0,button_1
CONTROL win,@BUTTON,"Button 2",10,190,150,40,0,button_2
CONTROL win,@BUTTON,"Button 3",160,200,150,25,0,button_3
CONTROL win,@BUTTON,"Button 4",160,190,150,40,0,button_4
CONTROL win,@BUTTON,"Button 5",310,200,150,25,0,button_5
CONTROL win,@BUTTON,"Button 6",310,190,150,40,0,button_6
CONTROL win,@STATIC,"Move the mouse over buttons 1,3 and 5.",10,140,300,35,@CTEDITLEFT,
STATIC_1
CONTROL win,@radiobutton,"This is a radio button",10,300,300,25,0,radio_1
CONTROL win,@checkbox,"This is a checkbox",10,350,300,25,0,check_1
CONTROL win,@BUTTON,"2nd screen",10,400,150,25,0,button_new
CONTROL win,@BUTTON,"Exit",770,400,100,25,0,button_exit

SETFONT win,"Tahoma", 8, 400, 0, button_5
SHOWWINDOW win,@SWHIDE,button_2
SHOWWINDOW win,@SWHIDE,button_4
SHOWWINDOW win,@SWHIDE,button_6
```

Store the co-ordinates and window details of the controls we want to keep track of.

In this example I want to track mouse over for button_1,3 & 5, a radio button, a checkbox & a static so we now use the following commands:

```
MouseOverControl(win,button_1,"win","B")
MouseOverControl(win,button_3,"win","B")
MouseOverControl(win,button_5,"win","B")
MouseOverControl(win,radio_1,"win","R")
MouseOverControl(win,check_1,"win","C")
MouseOverControl(win,static_1,"win","S")
```

Notice that the window description (in quotes) is the same for all the controls we want to keep track of.

Start a timer so we can keep checking to location of the mouse...

```
STARTTIMER win,100

WAITUNTIL win=0
```

StopTracking - MUST use this to delete the pointers used for tracking controls & windows before exiting.

```
StopTracking()

END
```

In the timer section of the handler for the window...

```
    CASE @IDTIMER
```

Always use TrackWin command first - tells you if the mouse is over the window BUT it also checks to see if the window has moved, if so it updates the controls with their new co-ordinates.

You can just use - TrackWindow(win,"win") on it's own if you do not want to do anything whilst the mouse is over the window.

```
        IF TrackWindow(win,"win")

            SETCAPTION win,"Mouse is Over window"

        ELSE
            SETCAPTION win,"Mouse is Not Over window"

        ENDIF
```

You could just use TrackWindow(win,"win") on It's own.

Notice that the window description (in quotes) is the same as the ones used in the MouseOverControls command earlier. They must be the same otherwise the controls will not be tracked.

Now we can track if the mouse is over a control or not using these commands...

```
    IF MouseOver(win,static_1,"win") = 1

        SETSIZE win,10,140,700,35,static_1
        SETFONT win,"Tahoma", 22, 400, 0, static_1

    ELSE

        IF MouseMovedOff(win,static_1,"win")
            SETSIZE win,10,140,300,35,static_1
            SETFONT win,"Arial", 10, 400, 0, static_1
            setcontroltext win,static_1,"Move the mouse over buttons 1,3 and 5."
        ENDIF

    ENDIF


    IF MouseOver(win,radio_1,"win") = 1

        SETFONT win,"Tahoma", 22, 400, 0, radio_1

    ELSE

        IF MouseMovedOff(win,radio_1,"win")
            SETFONT win,"Arial", 10, 400, 0, radio_1
        ENDIF

    ENDIF
```

For a working example see the example program included with this help file.

Multiple windows

Working with more than one window in a program.

Most likely, you will have more than one window in your program, so what options could you use whilst using the MouseOverControls command?

Option 1:

```
        CASE button_two Button to open Screen 2

            IF @NOTIFYCODE = 0

              STOPTIMER win Stop tracking of screen 1
              SHOWWINDOW win,@SWHIDE
              Screen2()
              SHOWWINDOW win,@RESTORE
              STARTTIMER win,100 Start tracking of screen 1 again

            ENDIF
```

Stoping the timer for screen 1 before showing screen 2 stops the tracking of controls on screen 1.

You need to re-start screen 1's timer again when screen 2 is closed so tracking can begin again for screen 1.

Screen 2 has it's own timer for tracking it's controls.

Option 2:

```
        CASE button_two Button to open Screen 2

            IF @NOTIFYCODE = 0

              Screen2()

            ENDIF
```

Screen 2 will open, and both screen 1 & 2 now have their controls tracked.

Note:- If screen 2 is overlayed on screen 1, screen 1's controls will still "react" when the mouse is over them even though you cannot "see" them.

Option 2 is best where the screens are NOT over or partly over each other.

***Option 1 is the best way to go in most circumstances*.**

FAQ

Frequently Asked Questions.

Does it work for any type of control?
Yes.

Does it work for more than one window in a program?
Yes.

Can I re-use a control in another window?
Yes - if you use Button_1 in window 1, you can use Button_1 again in window 2
BUT! you must give different window names for them.

In window 1:

```
MouseOverControl(win,button_1,"win","B")
```

And in window 2:

```
MouseOverControl(win2,button_1,"win2","B")
```

Can I use the same window description (shown in quotes above) for more than one window?
No, each one must be different.

Do I have to use the same window description name as the window it is in for these control commands?
No, you can call them anything you want, but to avoid confusion I would recommend it as shown above.

Do I have to use the MouseOver Control for every control I want to keep track of?
Yes.

Do I have to use a timer?
Yes, otherwise you would not know where the mouse position is.

What speed should I set the timer to?
I would recommend setting the time to 100 milliseconds, you can modify this but it needs to be a reasonably fast timer. **So unless it causes you problems, I would stick to this speed.**

```
STARTTIMER win,100
```

Do I have to use the TrackWindow command?
Yes, always - without this the controls would stop responding when the window is moved or re-seized.

Do I have to use the same window description that I used for the MouseOverControls command?
Yes, otherwise the controls will not be tracked.

Example:

```
        MouseOverControl(win,button_1,"win","B")
        MouseOverControl(win,button_2,"win","B")
        MouseOverControl(win,button_3,"win","B")
```

The TrackWindow command would be

```
        TrackWindow(win,"win")
```

Do I have to make the TrackWindow command do anything?
No, you can just add the command in the timer section.

Example 1:

```
        TrackWindow(win,"win")
```

Keeps updating the controls with their coordinates just in case the window was re-sized or moved.

Example 2:

```
      IF TrackWindow(win,"win")    (Keep doing the following)

        SETCAPTION win,"Mouse is Over window"

      ELSE

        SETCAPTION win,"Mouse is Not Over window"
        ENDIF
```

Allows you to do something if the mouse is over the window.

Do I have to use the MouseOver check and what does that mean?
Yes, When you use the Mouse over command like this:

```
        IF MouseOver(win,static_1,"win")  = 1    (Do the following only once)

           SETSIZE win,10,140,700,35,static_1
           SETFONT win,"Tahoma", 22, 400, 0, static_1

        ELSE.........
```

It means **do this one time only** when the mouse has moved over the control.

Note, you must use the **MouseMovedOff** command when checking
if MouseOver is equal to 1 or greater.

If you use:

```
        IF MouseOver(win,static_1,"win") > 0    (Keep doing the following)

           SETSIZE win,10,140,700,35,static_1
           SETFONT win,"Tahoma", 22, 400, 0, static_1

        ELSE.........
```

The MouseOver section will keep repeating the instructions that follow until you move off the control.

So the return value for the MouseOver command means:

The mouse has moved over the control   = 1   **(Do the following once)**
 OR the mouse remains over the control = 2   **(Keep repeating the following)**
 OR the mouse remains over the control > 0   **(Do the following once then keep repeating).**

Is the include file free to use and distribute?
Yes, of course.

Disclaimer

Disclaimer.

The MouseOverControls.inc file & the ButtonTest example code are provided "as is".

Whilst this code seems to work, you agree and accept to use it at your own risk and I am not responsible should it not be fit for purpose or as described or cause unforeseen problems for you or your code.

Thank you.