asMonitor - Help

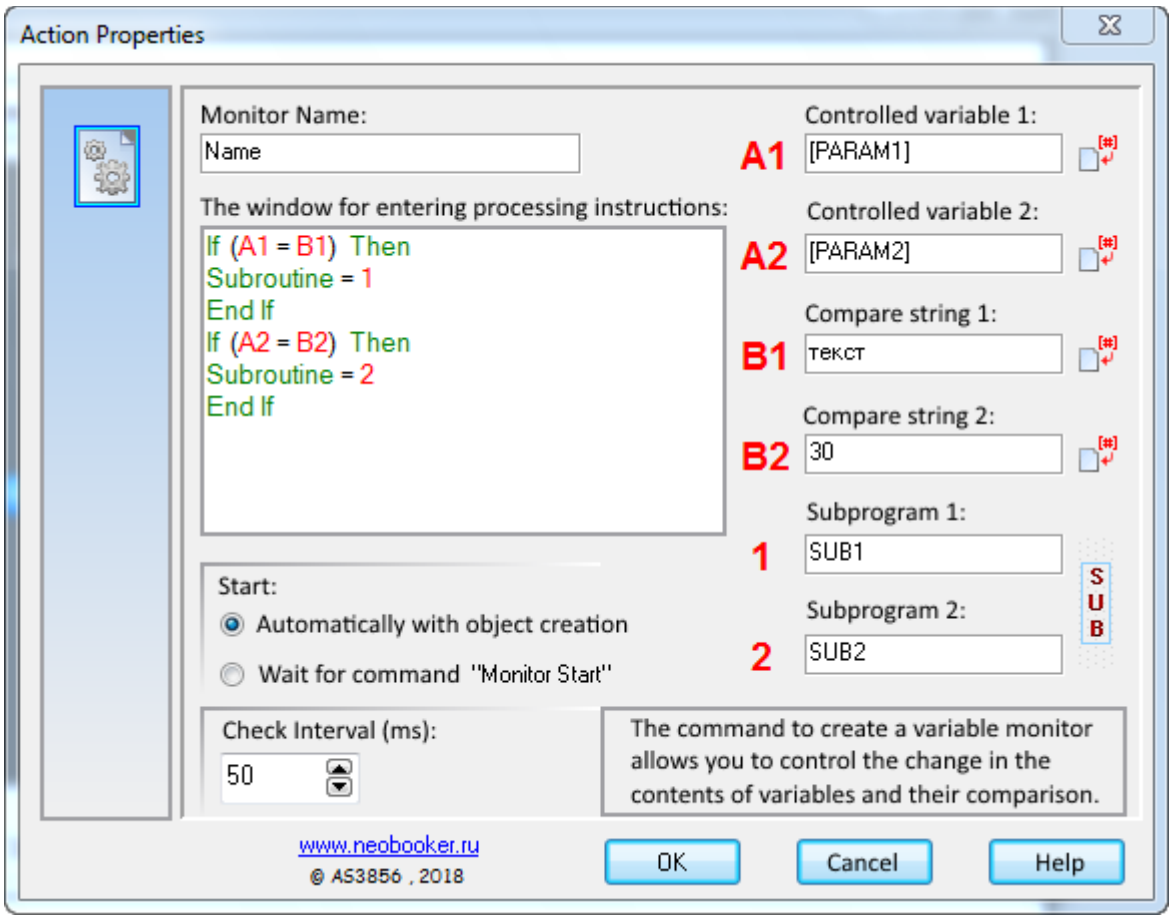
About the pluginThe plugin is designed to control (monitor) variables, while monitoring is carried out in a separate thread from the main NeoBook code and does not interfere with other tasks that are carried out in the NeoBook project. An example is the control of the global variable NeoBook [Time] for time control, while in performing other tasks this control will not interfere with the main code. (We create a project for some testing, set the time for which it is necessary to pass the test, look at the start time of the test, calculate the end time of the test and set this time as a comparison line (compare the current time with the estimated time) - when the estimated time is reached, a subroutine is executed (for example testing stops - an approximate simple plan for such a project).

Monitor creation

asMonitorCreate = Create an invisible Monitor object to control the selected variables.

asMonitorCreate - commands to create an invisible Monitor object to control selected variables.

Command Code Helper Window:



Monitor name - any name consisting of letters of the Latin alphabet

Controlled variable 1 - Select a variable for which it is necessary to control the contents. It can be any global or local (user) variables. In the variable recording window, the variable should be written (in square brackets).

Controlled variable 2 - It is filled in the same way as controlled variable 1.

Comparison string 1 - The line for comparison is entered here. If you want to control whether the content of the variable has changed, then we enter this variable here. If you want to compare the contents of the variable with something, then we enter the comparison string here. Comparison line 1 can be used to compare both controlled variable 1 and to compare controlled variable 2.

Comparison string 2 - It is filled in the same way as comparison line 1.

Subprogram 1 is one of the subprograms executed when a specified condition is fulfilled in the window for processing instructions.

Subprogram 2 is one of the subprograms executed when a specified condition is fulfilled in the window for processing instructions.

Start - the choice to start the monitor.

Check interval (ms) - the interval of the time interval for checking the status of the contents of variable 1 or variable 2 (or both, depending on conditions). Time in milliseconds. The minimum interval time is 50 ms. the maximum is 60,000 (1 min). Under difficult (branched) conditions, I recommend setting at least 100 ms (10 times per second) for successful execution of the script recorded in the input window for processing instructions.

The window for entering processing instructions

Designed to enter the conditions under which the subroutines will be executed. The plugin uses an interpreter from VBScript - a programming language designed to create scripts. It is a subset of the Visual Basic language and is widely used when creating scripts in Windows operating systems. VBScript (VBS) is a simplified version of the usual Visual Basic syntax. To facilitate code entry, writing was as easy as possible, leaving only the conditions directly.
All other commands required during normal script writing (error handling, indicating a function, ending a function) are executed automatically without user intervention.

As in NeoBook, reserved words are used to write conditions **If, End If, Then** (new for non-bookers), as well as **and, or, xor** and others more: http://www.nestor.minsk.by/kg/2000/02/kg00216.html https://msdn.microsoft.com/ru-RU/library/windows/desktop/dd409611(v=vs.120)

and operators (= , < , > , <= , > = and others). All processing instructions (conditions).

fit into the window line by line (the same as in NeoBook), while syntax highlighting of the main keywords is performed.

In addition to the keywords, the word **Subroutine** is added as a pointer to the subroutine being executed. Each field for entering variables, comparison strings, or subprograms is identified by an identifier.

(**A1 A2 B1 B2 1 2**)

An example of writing a condition:

```
If (A1 = B1) Then      // if (A1 is equal to B1) then
Subroutine = 1         // execute routine 1
End If                 // be sure to end the condition
```

write the following (second) condition

```
If (A2 = B2) Then      // if (A2 is B2) then
Subroutine = 2         // execute routine 2
End If                 // be sure to end the condition
```

It should be borne in mind that if any condition (first or second) is met, monitoring will be stopped and to resume it (depending on the tasks being solved), you must use the asMonitorStart plug-in command.

You can record more complex processing conditions, for example:

```
If (A1 <> B1) or (A2 <> B2) Then     // if (A1 is not equal to B1) or (A2 is not equal to B2)
Subroutine = 1                       // execute routine 1
End If                               // be sure to end the condition
```

You can write conditions for only one variable, or monitor and process only variables without comparison lines, for example like this:

```
If (A1 <> A2) Then     // if (A2 is not B2) then
Subroutine = 2         // execute routine 2
End If                 // be sure to end the condition
```

Window for entering processing instructions (numbers and strings)

The plugin can compare the contents of variables in a numeric or string expression. The determination of a number or a string occurs automatically. If during the operation of the monitor the contents of the controlled variable changes (string or number), respectively, and the comparison will change. For example, if the value of the controlled variable is equal to 20, and 20.0 is in the comparison line, then when comparing the variable and this line, they will be equal (like numbers). If the value of the controlled variable has changed to text, and there is a number in the comparison line, then the comparison will occur as for a line. For simple comparisons, it is sufficient to use comparison operators (= <>), and for complex string comparisons, you can use the string comparison function **StrCompare**, you can use other search and compare functions (it is even possible to search for characters in a string.) For such actions, you will have to study VBS string functions (e.g. https://vbsabout.wordpress.com/2013/02/25/workwithtext/ or http://scriptcoding.ru/2013/06/21/strokovyje-funkcii-vbs/)

Comparing Strings Using the StrComp Function

The function allows you to compare strings with and without case sensitivity of string characters. This is set by the parameter **vbTextCompare** (case-insensitive) or **vbBinaryCompare** (case-sensitive - you can not write it because by default the strings are case-sensitive).

Example 1:

```
If StrComp (A1, B1, vbTextCompare) = 0 Then // case insensitive comparison
SubRoutine = 1                              // subroutine 1 will be executed
End If                                      // be sure to end the condition
```

If you compare BIRCH and birch - then they will be equal in this condition.

Example 2:

```
If StrComp (A1, B1) <> 0 Then               // Case-sensitive comparison
SubRoutine = 1                              // subroutine 1 will be executed
End If                                      // be sure to end the condition
```

If you compare BIRCH and birch - then they will not be equal in this condition.

Comparing String Lengths Using the Len Function

The function allows you to calculate the length of a string.

Example:

```
If Len (A1) = Len (B1) Then // If The Length Of The String A1 Is Equal To The Length Of The String B1
SubRoutine = 1              // subroutine 1 will be executed
End If                      // be sure to end the condition
```

Start and stop commands

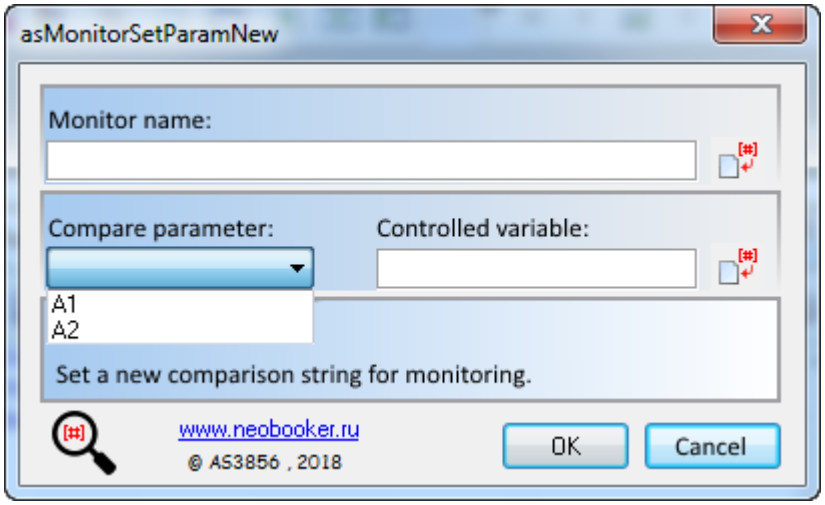
asMonitorStart = Start monitoring variables, if it was not started when creating the object.

asMonitorStop = Stop monitoring variables.

asMonitorDelete = Delete the Monitor object.

Setting variable and string to compare

asMonitorSetParamNew = Set the variable to monitor.



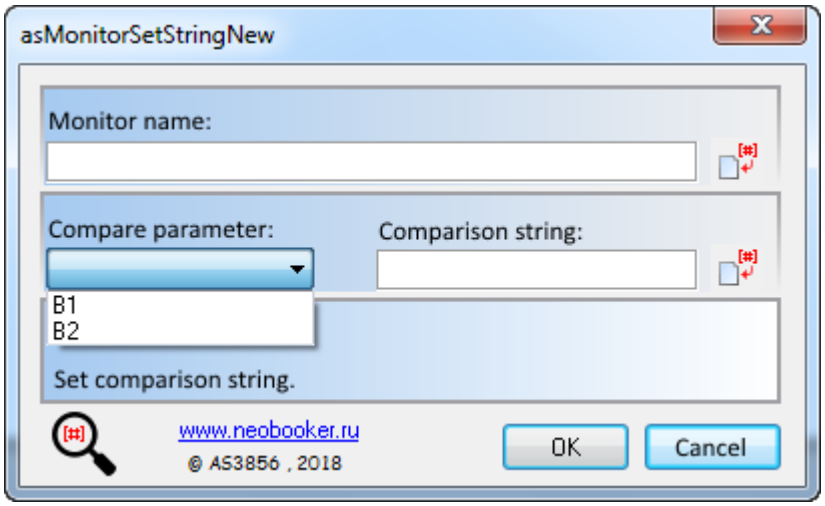
Using the command, you can replace the previously controlled variable with another variable. It should be borne in mind that in this case the comparison conditions remain and the compared string remains the same.

Monitor name - The name of an existing asMonitor object.

Compare parameter - select one of the identifiers that will point to the variable (see **asMonitorCreate**).

Controlled variable - select and enter the variable (in square brackets) that should be controlled.

asMonitorSetStringNew = Set a new comparison string for monitoring.



Using the command, you can replace the previously used comparison string with another comparison string. It should be borne in mind that in this case the conditions of comparison remain and the compared variable remains the same.

Monitor name - the name of an existing asMonitor object.

Compare parameter - select one of the identifiers that will point to the comparison string (see **asMonitorCreate**)

Comparison string - comparison string with which the variable will be compared.