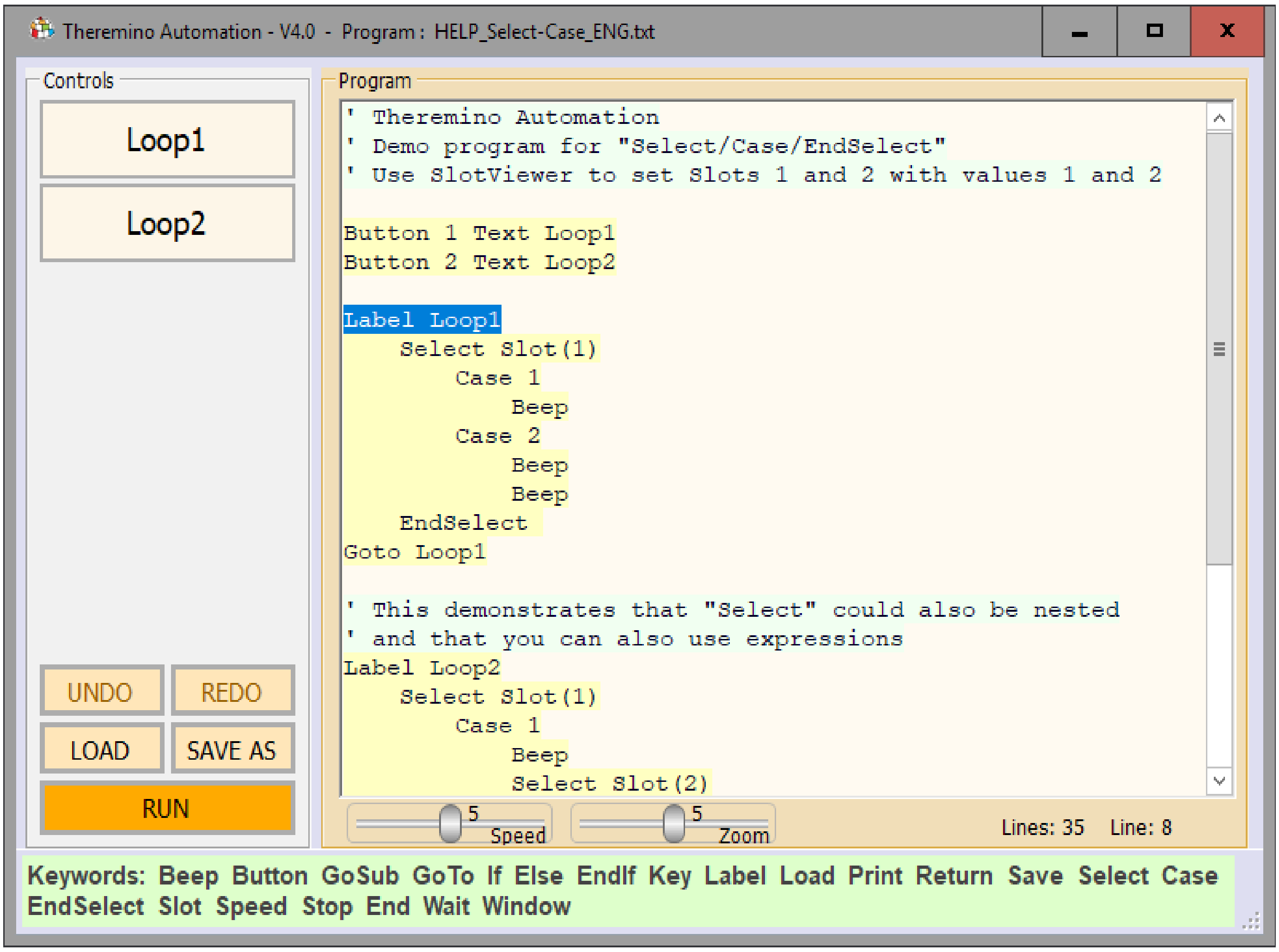
Theremino System


Theremino Automation - V4.0 - Program : HELP_Select-Case_ENG.txt

```
' Theremino Automation
' Demo program for "Select/Case/EndSelect"
' Use SlotViewer to set Slots 1 and 2 with values 1 and 2

Button 1 Text Loop1
Button 2 Text Loop2

Label Loop1
    Select Slot(1)
        Case 1
            Beep
        Case 2
            Beep
            Beep
    EndSelect
Goto Loop1

' This demonstrates that "Select" could also be nested
' and that you can also use expressions
Label Loop2
    Select Slot(1)
        Case 1
            Beep
            Select Slot(2)
```

Lines: 35 Line: 8

Keywords: Beep Button GoSub GoTo If Else EndIf Key Label Load Print Return Save Select Case EndSelect Slot Speed Stop End Wait Window

Theremino Automation V4.x

Preconditions

We simplified and minimized everything possible to facilitate those who have no experience in programming.

This language is only suitable for short programs. Typically small repetitive tasks such as, for example, read the value of a slot, and calculate the temperature, and then write the calculated value into another slot.

They are also possible complex operations such as play audio and video, or surf the internet. With a single instruction performing tasks that in other languages require specialized knowledge and many pages of code.

Automation of Theremino limits.

```
Keywords: Beep Button GoSub GoTo If Else EndIf Key Label Load Print Return
          Save Select Case CaseElse EndSelect Slot Speed Stop End Wait Window
```

In Automation instructions are few, about twenty. No classes are types, structures, and none of the complex mechanisms of the classic programming languages.

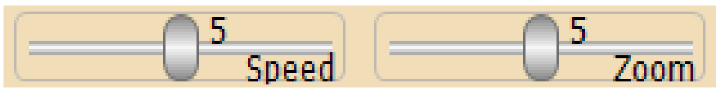
So, for complex tasks, you have to switch to more powerful programming environments (such as Visual Studio). As a guideline you can count lines, beyond the one or two hundred lines it is better to abandon Automation.

Theremino Automation is an interpreted language.

In languages "interpreted", unlike those "compiled", the instructions are not pre compiled, but are read, character by character, and interpreted during the execution itself.

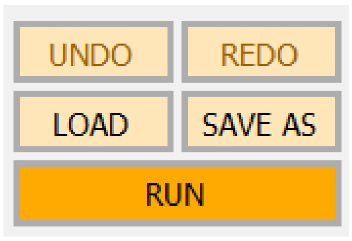
The execution of an interpreted language is therefore slower. In our case, the instructions are on average ten times slower than the same instructions written in Visual Studio (Csharp, C ++ or VBNET).

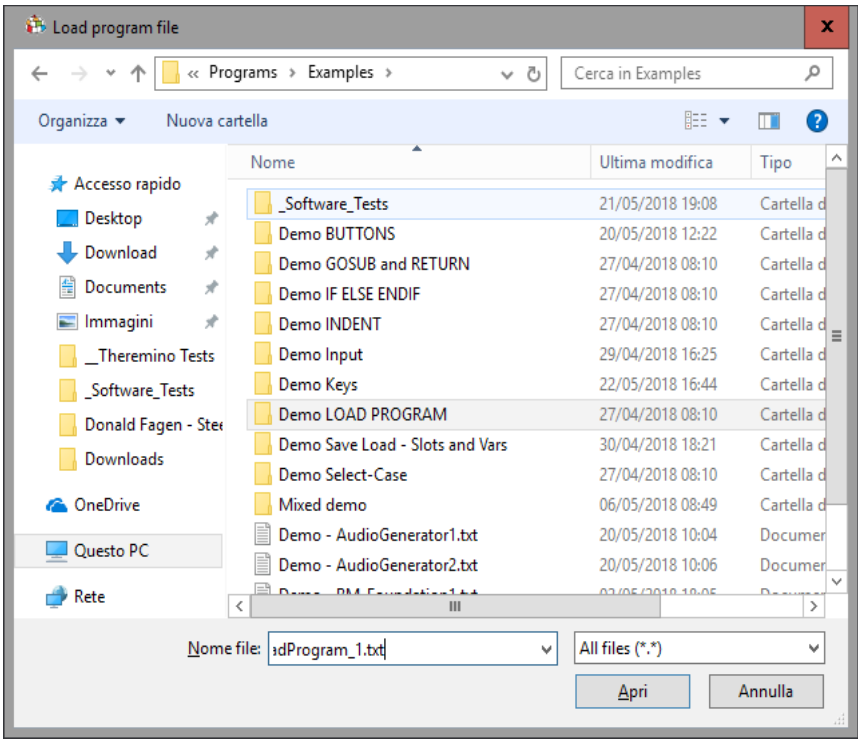
Speed is still abundant for simple tasks it is intended for this language. So much so that we had to add the ability to slow it down further, even thousands of times, to facilitate the understanding of what is happening.



Load, edit and save programs

Using the LOAD button opens a window that lets you choose the programs to load.





Nome	Ultima modifica	Tipo
_Software_Tests	21/05/2018 19:08	Cartella d
Demo BUTTONS	20/05/2018 12:22	Cartella d
Demo GOSUB and RETURN	27/04/2018 08:10	Cartella d
Demo IF ELSE ENDIF	27/04/2018 08:10	Cartella d
Demo INDENT	27/04/2018 08:10	Cartella d
Demo Input	29/04/2018 16:25	Cartella d
Demo Keys	22/05/2018 16:44	Cartella d
Demo LOAD PROGRAM	27/04/2018 08:10	Cartella d
Demo Save Load - Slots and Vars	30/04/2018 18:21	Cartella d
Demo Select-Case	27/04/2018 08:10	Cartella d
Mixed demo	06/05/2018 08:49	Cartella d
Demo - AudioGenerator1.txt	20/05/2018 10:04	Documen
Demo - AudioGenerator2.txt	20/05/2018 10:06	Documen

Nome file: adProgram_1.txt

After loading the program you run it with RUN.

Edit and save programs

Any changes you make to a program is saved automatically, so there is no need to remember to save it, before closing the application Automation, or load another program.

When you re-open the program will be exactly as you left it the last time.

So if you do not want to lose the original, before changing a program you should save it with a name, using the SAVE AS button.

If needed, you can recover the original examples, by copying them from the "Programs - Copy" folder, or you could retrieve it from the original ZIP file which you downloaded from the ***theremino site***.

Simple programs to start

About the program for the first time might find the simple examples that are in the "Simple Programs" folder.

The best way to start is to load these examples, and follow the instructions one at a time, with the "Single Step" of the "Debug" window button (to open it using the right mouse button and select Debug).

In the Debug window you are already present some lines that show the value of variables, of the slot and expressions. However, it is also easy to add new ones.

Pressing repeatedly Single Step, you can see the values change and understand the progress of the program, and how to run the calculations and assigned values.

Here is an example of a simple program

```
v1 = 10
v2 = 20
v3 = v1 + v2
v4 = v3 * 12
Slot (1) = Sqrt(v4)
Slot (2) = Rnd * 1000
```

Note that, after performing the last line, the program starts automatically from the first line. In case you want to stop it you should add a "Stop" line at the end.

Here is a second example

```
v1 = Rnd
If v1 > 0.5
     Beep
     Print ""
else
     Print "The v1 value is : " + v1
EndIf
```

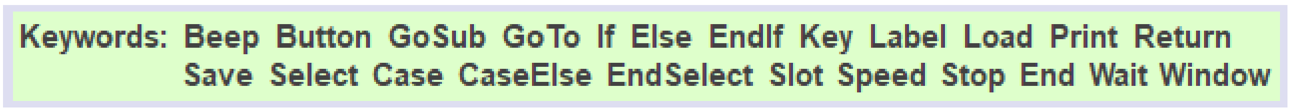
In this example the Rnd statement assigns, to the variable v1, a random value between 0 and 1. Then emits a sound if it is greater than 0.5, otherwise outputs its value.

- - - - -

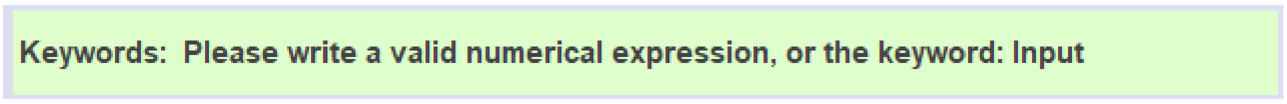
See also other examples who are in the "Simple Programs" folder

Keywords

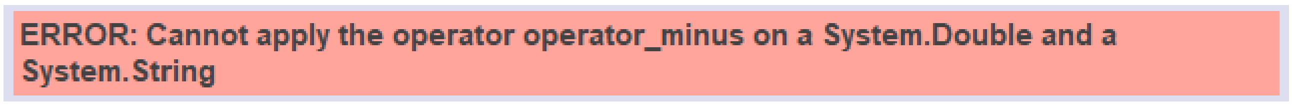
The keywords are only about twenty, easy to remember and easy to use.


Keywords: Beep Button GoSub GoTo If Else EndIf Key Label Load Print Return
Save Select Case CaseElse EndSelect Slot Speed Stop End Wait Window

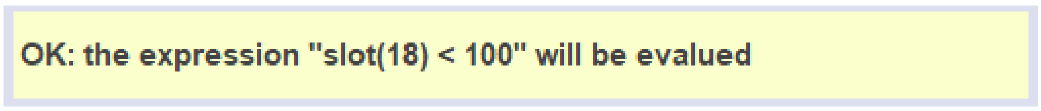
The bottom bar of the application displays the keywords that can be used.


Keywords: Please write a valid numerical expression, or the keyword: Input

While writing, the bottom bar shows suggestions to complete the instructions properly.


ERROR: Cannot apply the operator operator_minus on a System.Double and a System.String

Not all errors are identified and explained, we are not Microsoft, we are few and time is limited.


OK: the expression "slot(18) < 100" will be evalued

However, the bottom bar is a good help for beginners, and each new version works a bit 'better.

- - - - -

On the following pages the keywords will be explained in detail.

Note that they are listed in order "almost" alphabet (some words are grouped together because they like each other).

The key words one by one

Beep

This statement gives the basic sound established by the operating system. The sound is not adjustable, but can also be useful to point out to particular moments of the program.

To issue specific sounds such as a siren or the click of the button, you can use the LOAD instruction (see how you are using the following pages).

To emit complex sounds and adjustable, you should use other applications (from theremino collection) and controlled by Automation, with the SLOT statement.

For example, using the ***Audio Generator*** you could control the sounds in frequency and amplitude, as well as choose the output device. And also choose between sounds sinusoidal, triangular, ramp, square waves and white noise.

But with the ***Sound Player*** you could play songs by reading them from a file with variable speed, forward or backward and additional effects.

Finally, the application ***Theremin Sinth*** provides a complete audio synthesizer capable of playing with every sound type, from organ to guitar, etc ..

- - - - -

To control external applications with automation you write the check values in the slot.

For example for Audio Generator you could use the Slot one for the waveform, the two for the frequency, the three for the amplitude and the four for the Pan (position in stereo).

See also the examples

"Demo - AudioGenerator1" and "Demo - AudioGenerator2"

which are located in the folder "Programs / Examples"

Button

The buttons appear on the left side of the window and you can press them with the left mouse button, or a finger when using a touch screen.

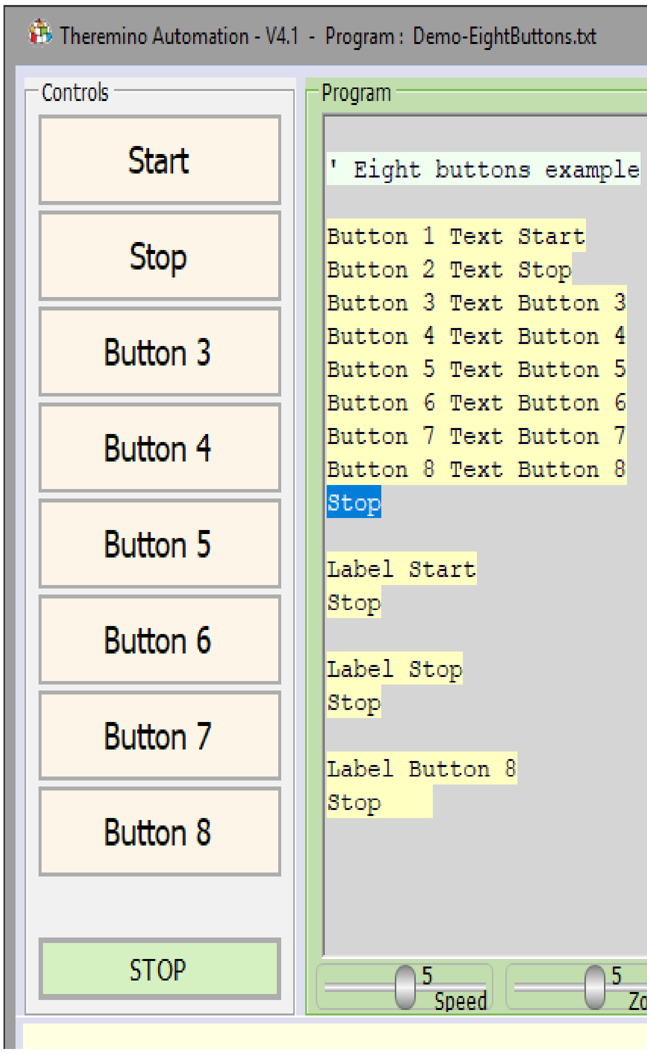
You can bring up to eight buttons and determine, for each of them, the text to display.

If there is a LABEL instruction with the corresponding text, then press the button, the execution will continue from the LABEL program online.

The jump to LABEL is GOSUB type. So, if you encounter a RETURN, the execution will resume from where it left off. Otherwise it will continue without returning.

If the text is more than one word then for the corresponding LABEL you can also use only the first word. One can also enclose the text with quotes, and consider it one word. If the text is very long, it will use a smaller font and, if necessary, the text will be written on several lines.

When working in full screen, active buttons are displayed on top of any image or video.



```
' Eight buttons example

Button 1 Text Start
Button 2 Text Stop
Button 3 Text Button 3
Button 4 Text Button 4
Button 5 Text Button 5
Button 6 Text Button 6
Button 7 Text Button 7
Button 8 Text Button 8
Stop

Label Start
Stop

Label Stop
Stop

Label Button 8
Stop
```

To make an invisible writing a blank text button (only two double quotes). To make it visible but no text, you write a space between double quotes.

To view all eight window buttons should be fairly high, otherwise late down buttons are not displayed.

The SLOT statement connects the button at a Slot. So, in this example: **`Button 1 Slot 6`**, if the Slot value exceeds 500, then the button is pressed.

- - - - - -

See also the examples in the "Demo BUTTONS"

Goto, Gosub, Return

The GOTO and GOSUB statements blasting the execution of the program to the mark with a LABEL (label) and with an identifying text, composed of alphanumeric characters.

So the instruction `Goto xxx` jumps to the line indicated by `Label xxx`.

And the instruction `Gosub yyy` jumps to the line indicated by `Label yyy`.

If there is no corresponding label, then the GOTO and GOSUB statements have no effect and the execution continues as if there were.

The identifier can also be composed of several words, and these words can also be enclosed in double quotes (for clarity), for example, you could write `Label xxx yyy zzz`, or `Label "xxx yyy zzz"`.

Difference between Goto and Gosub

To put it in simple words, the GOTO is like a one-way trip, while the GOSUB also provides for the return.

So if you use a GOTO, the program jumps and continues from that position.

While if you use a GOSUB, the program jumps, performs some instructions and, when it encounters a RETURN statement, returns to the statement after the GOSUB.

```
...
Gosub Identifier1
...
...
...


Label Identifier1
...
Return
```

- - - - - -

See also the examples in the "Demo GOSUB and RETURN"

If, Else, endif

The "IF" and "ENDIF" instructions enclose an area to be executed only if the condition is true.

Each "IF" must end its "ENDIF" otherwise does not run and the execution continues as if there was not.

```
If Slot (1)> 500
     Instructions to execute
     only if the value of the slots 1
     is greater than 500
endif
```

The instructions "IF", "ELSE" and "ENDIF" contain two zones, one to run if the condition is true, and the other to run if the condition is false.

```
If Slot (1)> 500
     Instructions to execute
     only if the value of the slots 1
     is greater than 500
else
     Instructions to execute
     if it is less or equal than 500
endif
```

The IF statements (and all the other Automation structures) can be "nested" that is placed one inside the other. For example in the following example the BEEP is performed if all three slots are larger than 500.

```
If Slot (1)> 500
     If Slot (2)> 500
          If Slot (3)> 500
               Beep
          EndIf
     EndIf
EndIf
```

But the same result you could get even with AND instructions

```
If Slot (1)> 500 And Slot (2)> 500 And Slot (3)> 500
     Beep
EndIf
```

- - - - - -

See also the examples in the "Demo IF ELSE ENDIF" folder

Key

With the keyword "Key" is obtained by the execution of program parts, pressing keys on the keyboard.

The usable keys are as follows:
1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, Left, Right, Up, Down, Space, Esc, PageUp, PageDown, Home, End

The KEY statements usually are written at the beginning of the program and can be of two types GOTO or GOSUB. With GOTO The jump and never comes back. With GOSUB the program jumps back when encounters a RETURN.

```
...
Key 1 Goto Identifier1
Key 2 Gosub Identifier2
Stop

Label Identifier1
...
Stop

Label Identifier2
...
Return
```

When you press a key on the keyboard, the program is stopped, whatever he was doing, and jumps to the corresponding LABEL.

If you have set a KEY with RETURN, then the program is interrupted, jumps to executing lines after the LABEL, then meets the RETURN, and go back to doing what he was doing when he was interrupted.

To avoid accidentally trigger it while you write something, the Key do not work when you browse pages on the web, or when you are writing in an input box, or when you are writing in another application. Or in all cases when the application Automation is not selected, or is invisible, or is minimized.

- - - - - -

See also the examples in the "Demo Keys"

Label

The LABEL instruction (label) is a marker, used to give a unique name to a program line.

The mark may consist of any number of alphanumeric characters, also separated by spaces, as in the following examples:

```
Label 1

Label 123

Start Label

Label Stop

Label My Function

Label "MyFunction"

Label "My function with a long name"
```

The line labeled with the LABEL instruction, may be the target of the jump instructions: GOTO, GOSUB, and KEY BUTTON.

Here are examples:

```
GoTo MyFunction

GoSub MyFunction

Key 1 GoTo MyFunction

Key 1 GoSub MyFunction

Button 1 Text MyFunction
```

- - - - - -

See also the examples in folders
"Demo GOSUB and RETURN" and "Demo Keys"

Load (Applications)

In some cases it may be useful to start an application as many times as you run Automation. Often this application is Theremino_HAL or SlotViewer, or even both.

If the application is already active for a second time is not launched.

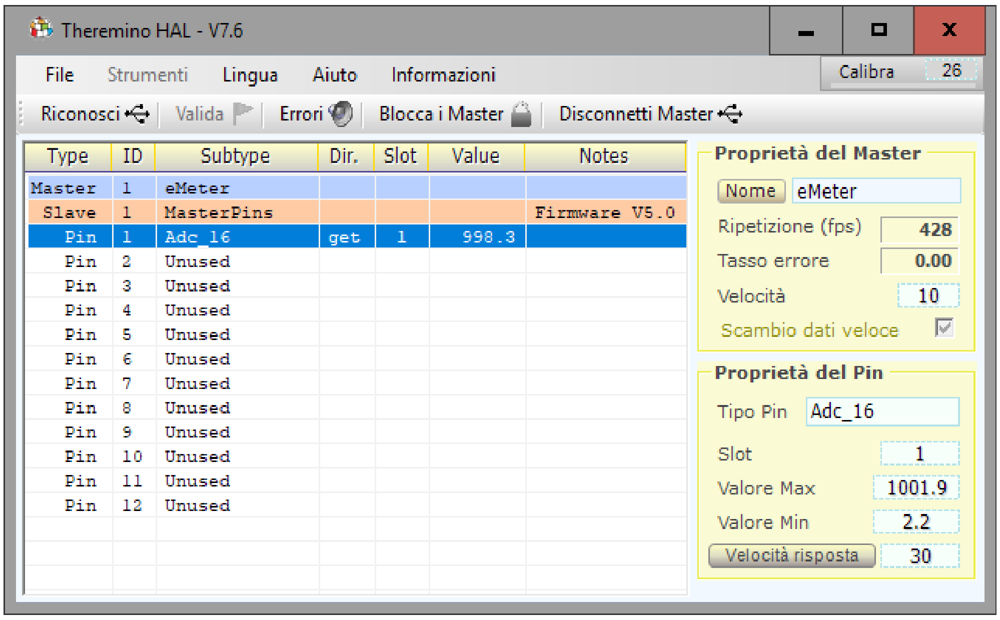
To upload their applications "xxxx.exe" file should be in the same folder of the Automation.exe file.

You could also give the full path, but should be in a fixed place in the system (not in the Automation folder), or by moving the Load Automation stop working.

If the application to launch is located in a sub-folder, you can add the part of the path you need, such as:
Load Theremino_HAL\Theremino_HAL.exe

If there are spaces in the path, you must add quotes
Load "Theremino HAL\Theremino_HAL.exe"

If the application to launch is in a parent folder, you can go up a folder with the colon. Eg:
Load ".. \ Theremino_HAL\Theremino_HAL.exe"



Type	ID	Subtype	Dir.	Slot	Value	Notes
Master	1	eMeter				
Slave	1	MasterPins				Firmware V5.0
Pin	1	Adc_16	get	1	998.3	
Pin	2	Unused				
Pin	3	Unused				
Pin	4	Unused				
Pin	5	Unused				
Pin	6	Unused				
Pin	7	Unused				
Pin	8	Unused				
Pin	9	Unused				
Pin	10	Unused				
Pin	11	Unused				
Pin	12	Unused				

Proprietà del Master

Nome	eMeter
Ripetizione (fps)	428
Tasso errore	0.00
Velocità	10
Scambio dati veloce	✓

Proprietà del Pin

Tipo Pin	Adc_16
Slot	1
Valore Max	1001.9
Valore Min	2.2
Velocità risposta	30

Load / Save Slots

The `Load Slots` instruction loads all the values of the slot (from slot 0 to slot 999), by reading them from the "_Slots.txt" file.

The "_Slots.txt" file contains the values of all the Slots, which had been previously saved with the instruction `Save Slots`.

With `Load Slots n1 n2` you load only from Slot n1 to Slot n2, included.

Instead of n1 and n2 you can also write variables (v1 to v9), or formulas, also complex. Note that the formulas of Load Slots instructions must not contain spaces, or give an error. To use spaces in these formulas you might enclose them with double quotes.

Here are some examples:

`Load Slots 20 30`	slots from 20 to 30 inclusive
`Load Slots v1 v2`	slots from 10 to 20 inclusive
`Load Slots 2*3 2+v2`	slots from 6 to 22 inclusive
`Load Slots 2 Pi*4`	slots from 2 to 13 inclusive

In the examples it is assumed that v1 is equal to 10, and v2 which is equal to 20.

In the last example the PI (Pi function with value 3.14159...), is multiplied by 4, and then rounded to the nearest integer (13).

Load / Save Vars

The `Load Vars` instruction load all the variables (v1 to v9), by reading from the "_Vars.txt" file.

The "_Vars.txt" file contains the values of all variables that had been previously saved with the instruction `Save Vars`.

- - - - - -

See also the examples in the "Save Load Demo - Slots and Vars" folder

Load (Images, Videos and Sounds)

With the LOAD instruction you can upload images, videos and sounds, they must be in the "Media" folder.

Image files can have the following extensions:
.jpg
.bmp
.png

Video files can have the following extensions:
.gif
.avi
.wmv
.mpg
.mp4
.mov

The sound files (and music) may have the following extensions:
.mp3
.wav

Here are some examples of statements that load these files
Load Image1.jpg
Load Video1.avi
Load "Science Picnic 2014.mp3"

After loading the video and audio, you could stop them, and make them invisible, with the following instructions:
`Load Stop` stops the execution of the video and audio.
`Load Hide` become invisible images and videos (the program reappears).

- - - - - -

See the example "Demo-ImagesVideoSounds.txt"

The following two examples show that the filename
can also be in a string variable (S1 to S9)
and that you can concatenate strings and numbers
"Demo - BM_Foundation1.txt"
"Demo - BM_Foundation2.txt"

Load (Program)

With the Load statement you can load a program by choosing it among those who are present in the Automation folder.

The name of the program to be loaded must be enclosed in double quotes.

The program file to load must be located in subfolders of the application Automation, starting from the "Programs" folder.

The program file to upload must end with the extension ".txt"

Here's an example:

```
Load "Examples\Demo LOAD PROGRAM\Demo-LoadProgram_2.txt"
```

The program that is uploaded will replace the current program and will continue running immediately, starting with the first line of the newly loaded program.

- - - - - -

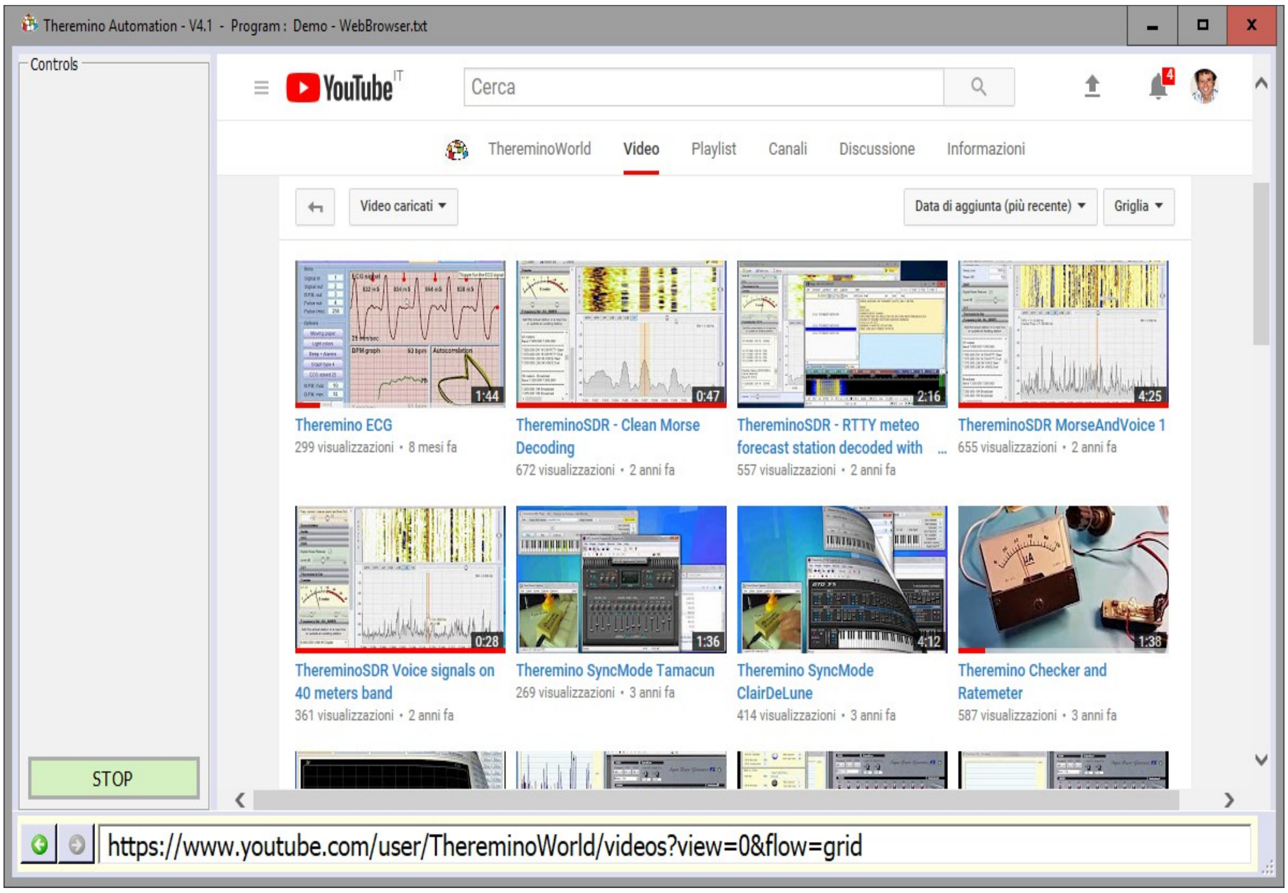
See also the examples in the "Demo LOAD PROGRAM" folder

Load (web address)

The LOAD instruction, followed by a web address, opens a web page.

Example: `Load http://www.google.com`

When writing the addresses in the LOAD instruction, the initial part http:// it's necessary.


Theremino Automation - V4.1 - Program : Demo - WebBrowser.txt
Controls
STOP
YouTube
Cerca
ThereminoWorld
Video
Playlist
Canali
Discussione
Informazioni
Video caricati
Data di aggiunta (più recente)
Griglia
Theremino ECG
299 visualizzazioni • 8 mesi fa
ThereminoSDR - Clean Morse Decoding
672 visualizzazioni • 2 anni fa
ThereminoSDR - RTTY meteo forecast station decoded with ...
557 visualizzazioni • 2 anni fa
ThereminoSDR MorseAndVoice 1
655 visualizzazioni • 2 anni fa
ThereminoSDR Voice signals on 40 meters band
361 visualizzazioni • 2 anni fa
Theremino SyncMode Tamacun
269 visualizzazioni • 3 anni fa
Theremino SyncMode ClairDeLune
414 visualizzazioni • 3 anni fa
Theremino Checker and Ratemeter
587 visualizzazioni • 3 anni fa

https://www.youtube.com/user/ThereminoWorld/videos?view=0&flow=grid

In this image you see a YouTube page, with videos of Theremino system.

Once you open a page it is possible to navigate anywhere on the web, using links found on web pages.

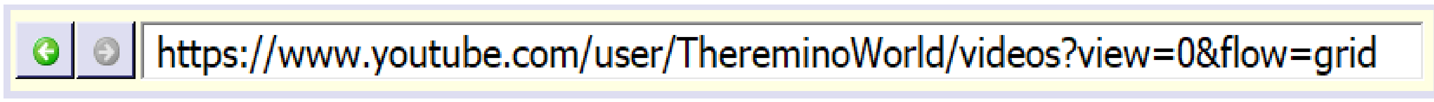
Alternatively, you can write a new address in the bar at the bottom, as explained on the next page.

Load (options for web pages)

The address bar

Writing an address in this bar, you can navigate to a new web site. When writing in this bar, you can also omit the http:// that precedes the address.

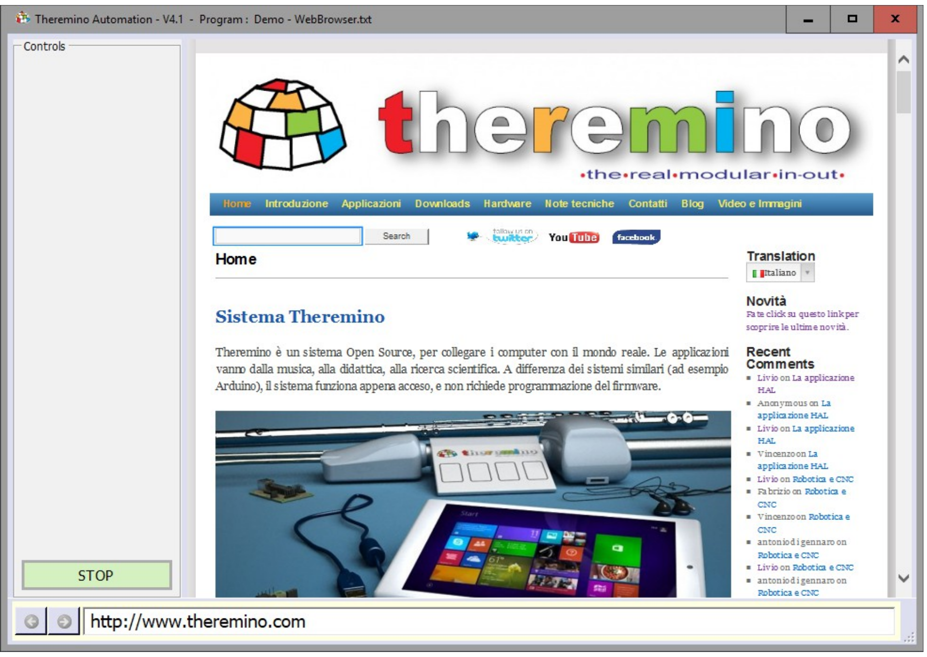
After writing the address, to begin the navigation to the new site, you must press ENTER.



https://www.youtube.com/user/ThereminoWorld/videos?view=0&flow=grid

The two buttons on the left are used to open the previous or the next page.

Zoom the page in and out



Theremino Automation - V4.1 - Program : Demo - WebBrowser.txt

Sistema Theremino

Theremino è un sistema Open Source, per collegare i computer con il mondo reale. Le applicazioni vanno dalla musica, alla didattica, alla ricerca scientifica. A differenza dei sistemi similari (ad esempio Arduino), il sistema funziona appena acceso, e non richiede programmazione del firmware.

Fate click su questo link per scoprire le ultime novità.

- Livio on La applicazione HAL
- Anonymous on La applicazione HAL
- Livio on La applicazione HAL
- Vincenzo on La applicazione HAL
- Livio on Robotica e CNC
- Fabrizio on Robotica e CNC
- Vincenzo on Robotica e CNC
- antoniodigennaro on Robotica e CNC
- Livio on Robotica e CNC
- antoniodigennaro on Robotica e CNC

http://www.theremino.com

In some cases it may be useful to magnify the page, for easier reading of the characters.

In other cases it may be useful to reduce its size, to avoid having to use the sliders too often.

To zoom the web page, you can use keyboard shortcuts, or the mouse wheel.

First of all you must click on the web page to be sure it is selected, otherwise the following commands will not work.

- Press Ctrl and "+" key, to enlarge the page.
- Press CTRL and the "-" button, to reduce the size of the page.
- Press CTRL and the ZERO button, to reset the normal magnification.
- Press CTRL and scroll the mouse wheel, to change the magnification.

Print

This statement prints, on the application bottom line, a numerical value or a character string.

After the word PRINT you can write any formula, even complex, composed with numerical values and character strings.

Examples:

`Print "Donald Duck"`	result: `Donald Duck`
`S1 = "PI ="`	
`S2 = Format (PI, "")`	
`Print S1 + S2`	result: `PI = 3.14159265358979`
`Print "Duck" = "Cat"`	result: False
`Print Mid("Duck", 4, 1) = "k"`	result: True

Usually you use this instruction to check the the program operation, to know the intermediate values of the calculations and to highlight that certain lines have been executed.

- - - - - -

To experiment with the Print instruction
use the example "Demo - Print.txt"

Save

Currently the SAVE instruction has only the ability to save slot and variables.

The instruction `Save Slots` saves the values of all slots (from 0 to 999), in the "_Slots.txt" file.

The instruction `Save Vars` saves all the variables values (v1 to v9 and s1 to s9), in the file "_Vars.txt"

- - - - -

For more info read the pages
Load/Save Slots and ***Load/Save Vars***

Select, Case, CaseElse and EndSelect

The Select-Case construct allows you to choose between a number of cases. In these examples could be only two "Case" but it could be dozens.

Obtaining the same operation with a series of "IF" would be more complex and the operation would be less noticeable.

```
Select v1
    Case 1
        Print 1
    Case 2
        Print 2
    CaseElse
        Print "Not 1 and not 2"
EndSelect
```

In this example, if the variable v1 is equal to 1 then the first "Case" is selected. If is equal to "2" then the second "Case" is selected.

The "CaseElse" is selected if all the preceding "Case" are not matched.

Our Select-Case implementation is particularly flexible. Almost all languages require that after the "Cases" there is a constant value, instead in Automation you can also write complex expressions.

In the next image you can see a simple example, only functions that read slot, but as addition and multiplication, and after the "Select" after the "Case", you can write expressions of any complexity.

```
v1 = 3
Select Slot(1)
    Case v1 * 2
        Print "This is executed when Slot(1) = 6"
    Case Slot(2) + 2
        Print "This is executed when Slot(1) = Slot(2) + 2"
EndSelect
```

WARNING: a terminal "EndSelect" is always required.

- - - - -

See also the examples in the folder "Demo Select-Case"

Select - Case (special features)

Our Select-Case implementation allows unusual comparisons, impossible in almost all programming languages, but very useful.

In practice you can compare everything, strings, numbers, boolean operators, and expressions with true or false results, permutating them as desired, in the Select and Case instructions.

For example, you could write "Select True", and the first "Case" with an expression that is "True", would be executed .

```
Select True
     Case Slot (1) <> 0 And s1 = "Is OK" And v1 = 3
          ...
EndSelect
```

If instead you wrote "Select False", the first "Case" with an expression that is "False", would be executed.

Alternatively, you could write "Select Slot (1) = 3" and the "Case True" would be executed, in case the Slot (1) was 3. Otherwise the "Case False" (or the "CaseElse") would be executed.

```
Select Slot (1) = 3
     Case True
          ...
     Case False
          ...
EndSelect
```

Could also write "Case Slot (2) = 5", and this would behave like a "Case True", only if the Slot (2) was worth 5. Otherwise it would behave like a "Case False".

You can therefore make all kinds of statements and choices, writing them in the way you prefer, and that seems more natural.

- - - - -

See in the folder "Demo Select-Case", the examples:

“Demo – Select-Comparations”

“Demo - Select-True-False”

Slots

This instruction reads and writes the Slots.

The slots are the center of the theremino-system communication, and who reads this page **it should** already know what they are.

Otherwise, we recommend ***read this section***, and maybe the whole ***communications page*** from start to finish.

Here are some examples of writing in a Slot

`Slot(1) = 12`	The value 12 is written in Slot 1
`Slot(2) = Rnd * 1000`	In the Slot 2 is written a random value from 0 to 1000
`Slot(3) = Sin(PI * 0.3)`	In the slot 3 is written the number 0.587785243988037

Examples reading from a Slot

```
v1 = Slot(1)
```

The value of slot 1 is read and assigned to the variable v1

```
If Slot(2) < 500
     Beep
EndIf
```

If Slot 2 is less than 500, than a sound is played

More complex expressions are also possible, the following expression writes the value 16 into Slot 3.

```
Slot (1) = 12
Slot (5) = 3
Slot(Slot(5)) = Slot(Sin(2)) + Slot(1) / 3
```

- - - - -

See also the examples in the "Demo Slots" folder

To view the values of variables and Slots,
you may want to open the Debug window, with the right mouse button.

Speed

Normally you adjust the program execution speed with the SPEED slider, explained in ***This Page***. But in some cases, you may need to impose a speed to the program, or to certain parts of the program.

Therefore, you can write the Speed instruction in the same program, and in this case the Speed slider is no longer considered.

Stop, End

These two statements are similar, both stop the program execution, but STOP is a kind of pause, while END is final.

With STOP the program stops, but continues to answer the PC keyboard (instruction **`Key n Goto/Gosub xx`**), Buttons (instruction **`Button n Text xx`**), and also to Slots (instruction **`Button n Slot nn)`**.

Instead, the statement END totally terminates the program execution. To start it again you will have to press the RUN button.

Wait

The program stops on the line of the WAIT instruction, and will continue in the following line, upon the occurrence of certain conditions.

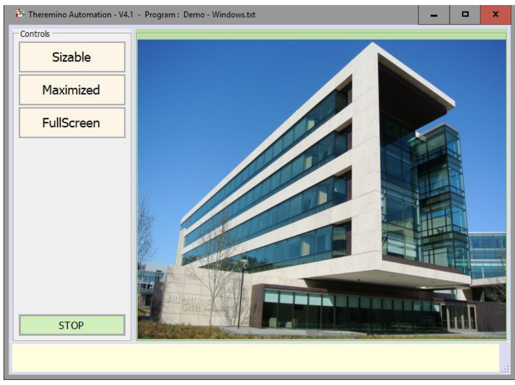
Here are some examples:

`Wait Seconds 1:53`	Waits for a second and 53 cents
`Wait Button "Button 1"`	Waits for the pressure of the Button 1 (if initialized)
`Wait Slot (4)> 500`	Waits for the Slot 4 value becoming greater than 500

- - - - -

See also the examples in the "Demo Wait" folder

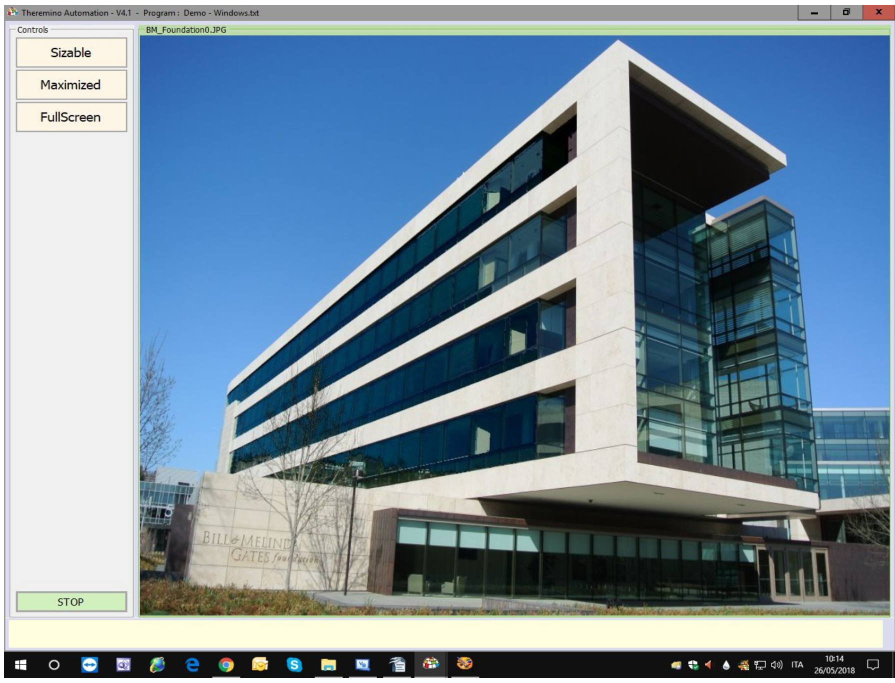
Window



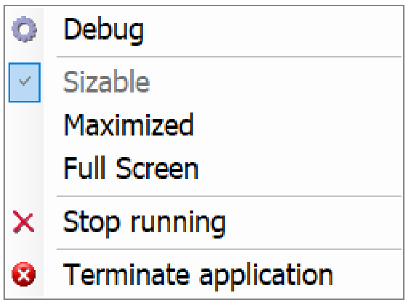
With the WINDOW statement you can change the size of the window, during program execution.

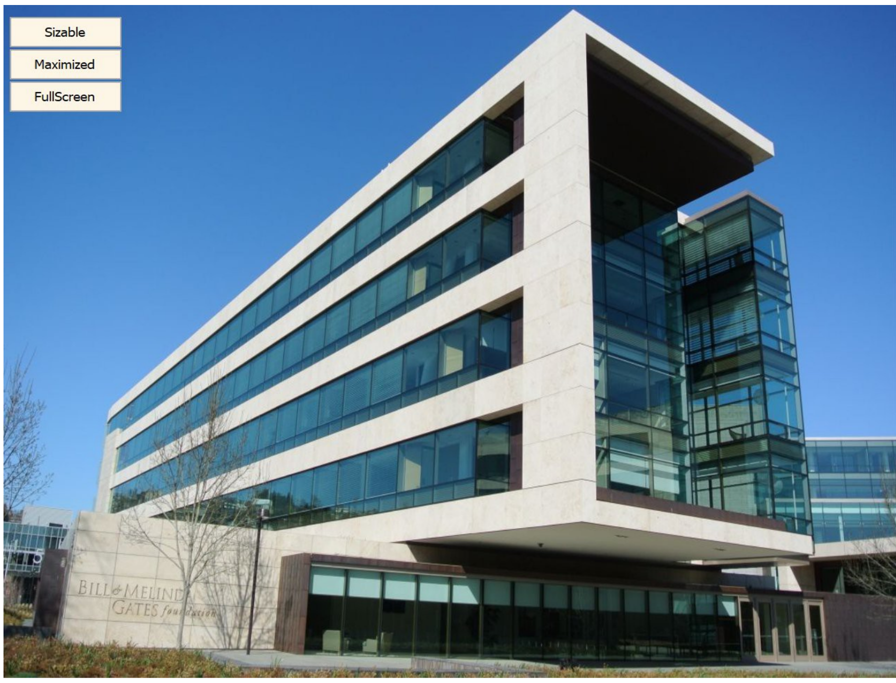
The WINDOW statement exists in three versions:

```
Sizable window
window Maximized
window Fullscreen
```



You can also choose these options manually, using commands ***menu of the application***, Which opens with right-mouse.





See also the example "Demo - Windows.txt"

Expressions and Functions

Automation also allows the use of complex expressions. In all the places where you could write a number or a character string, in its place you can write an expression.

Expressions may contain functions, operators, variables and constants. Some expressions, especially the IF and CASE constructs, can also contain conditions.

Functions

```
Slot(n) Sin(v) Cos(v) Asin(v) Acos(v) Tan(v) Atan(v) atan2(v, v)
Pow(v, v) Sqrt(v) Int(v) Round(v) Abs(v) Mid(string, index, len)
Len(string) Trim(string) Format(string, style) UCase(string)
LCase(string) WCase(string) Date(year, month, day)
Now  Today  Rnd  PI  Input  ElapsedTime
```

Operators and constants

```
And Or + - * /
```

the variables

`v1 v2 v3 v4 v5 v6 v7 v8 v9`	Numerical variables (with 16 decimal)
`s1 s2 s3 s4 s5 s6 s7 s8 s9`	String variables (use " " around the text)

constants

```
False True Reset Sleep
```

The conditions

```
Conditions are expressions that end with True or False.
True or False is determined by comparisons (> <=> = <= <>).
```

```
Given that this language does not type variables (it automatically
converts Booleans, Numbers and Strings when needed), in some cases
the ZERO could be considered False, and any other number
considered True.
```

- - - - -

Here are some examples of expressions and conditions

2 * 3 + 4 * Sqrt(16)	This is an expression, with result 22
`PI * 2`	This is an expression, with result 6.2831853 ....
`"Duck" = "Cat"`	This is a condition, with result False

For other examples of functions open the samples file "Demo - Print.txt"

Numerical functions

Pi - Returns the value of Pi Greek (3.14159265358979)

Sin (number) - Returns the sine of the number

Cos (number) - Returns the cosine of the number

Asin (number) - Return the arc sine of the number

ACOS (number) - Return the arc cosine of the number

Tan (number) - Return the tangent of the number

Atan (number) - Return the arc tangent of the number

Atan2 (number, number) - Arctangent extended to four quadrants

Pow (number, number) - Return the first number, elevated to the second

Sqrt (number) - Return the square root of the number

Abs (number) - Returns the absolute value of the number (always positive)

Sign (number) - Returns the sign of the number (-1, +1, or zero)

Int (number) - Returns the integer, rounded down

Round (number) - Returns the number rounded to the nearest

Rnd - Returns a random number between zero (inclusive), and one (not included).
The number is totally random because the Randomize function is used.

String functions

Mid (string, start, len) - Returns a string starting from "start", and long "len"

Len (string) - Returns the string length (number of characters)

Trim (string) - Returns the string without leading and trailing spaces

Ucase (string) - Returns the string with all uppercase characters

Lcase (string) - Returns the string with all lowercase characters

Wcase (string) - The first letter capitalized and the other tiny

Format (number, style) - Convert to String (see next page)

The “Format” function

The Format function (number, style) converts a number to a string.

In the "style" parameter you must provide a string that specifies how it should look the converted number.

`Format(1234.56, "000000,000")`	Result: "001234.560"
`Format(1234.56, "0.0")`	Result: "1234.6"
`S1 = Format(1234.56, "0")`	Result: "1235"
`S1 = Format(PI, "")`	Result: "3.14159265358979"

There are many other style options, such as the next two styles, that convert into scientific notation with one or two-digit exponent.

```
"0.000000E+0; -0.000000E+0"
"0.000000E+00; -0.000000E+00"
```

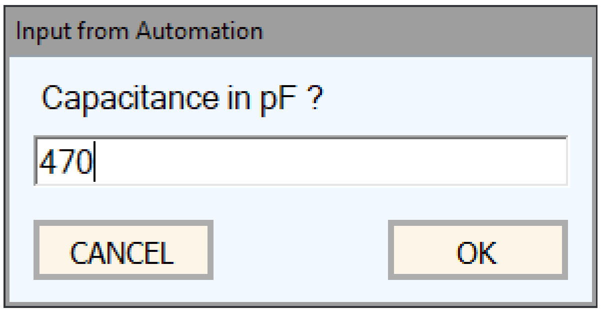
In this application (and in the whole scientific world), the point is always used for decimals, even in nations and operating systems that use the comma.

For a full explanation of all possible formats read ***This Page***.

The “Input” function

This function pauses the program and waits for the user to input a value.

The value can be a string of characters, or a number, depending on whether you use on the left of a string variable (s1..s9) or numeric (v1..v9).



If the user presses CANCEL the variable is not changed.

Examples:

```
S1 = Input "Write your input here"
V1 = Input "Write a number"
```

- - - - -

See also the examples in the "Demo Input" folder

The date and time functions

Date (Year, Month, Day) - Year, month and day converted to date

Now - Returns the date and time of the present moment

Today - Returns the date of the present moment

The functions that return a date (Date, Now and Today) may be used in formulas. You can add or subtract one from the other and you can also add or subtract with numbers. In this case the numbers represent days and the decimal part of the number represents fractions of days (which are not minutes or seconds but "fractions of days" ie, for example, "0.5" days are worth 12 hours).

The “ElapsedTime” function

ElapsedTime - Runners of the program.

This feature provides the seconds, since the program was started. The seconds include decimals to the tenths of a microsecond.

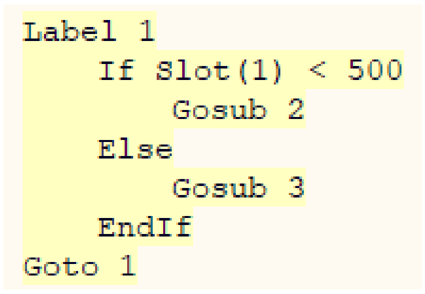
To measure a time interval (stopwatch), you can set the starting time in a variable (from v1 to v9) and then subtract it from the final time. As in the following example:

```
v1 = ElapsedTime
...
...
Print ElapsedTime - v1
```

If you hold up the program for a long time, the ElapsedTime value grows a lot, but not have to worry about that. Numeric rounding do not cause loss of precision, and microsecond continue to be valid, even if you keep running the program for many years.

To memorize time values it is best to use v1..v9, which are double-precision (16 significant digits), and not the Slots, that contain single-precision numbers (8 significant digits).

Auto indentation



```
Label 1
    If Slot(1) < 500
        Gosub 2
    Else
        Gosub 3
    EndIf
Goto 1
```

The indentation improves the visibility of the program, highlights the beginning and end of the structures, and makes it easier to write error-free software.

Therefore, the Automation application automatically indents programs, while writing them.

Get used to see the indented software is an important teaching aid.

It will become easier then indenting manually, also using programming environments that do not have the automatic indentation.

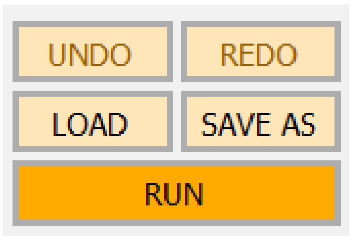
```
Label testAllSounds
    Wait Seconds 0.01
    If Slot(11) < 100
        If Slot(12) < 100
            If Slot(13) < 100
                If Slot(14) < 100
                    If Slot(15) < 100
                        If Slot(16) < 100
                            If Slot(17) < 100
                                If Slot(18) < 100
                                    Goto loop
                                EndIf
                            EndIf
                        EndIf
                    EndIf
                EndIf
            EndIf
        EndIf
    EndIf
Goto testAllSounds
```

- - - - - -

See also the examples in the "Demo INDENT" folder

The control buttons

The main functions are always available on the buttons, which can be pressed with the mouse, or your finger on a touch screen.



The most important is the RUN to start and stop the program.

Then there are the LOAD and SAVE AS buttons, which are used to load and save programs (see also ***This Page*** that explains how to edit and save programs).

The UNDO button is used to go back, if you have made changes to the program and want to eliminate them.

The REDO button reconstructs the changes eliminated by UNDO.

UNDO and REDO are part of the program's editing functions, but they are here to use them conveniently and repeatedly. The other editing functions are in the "Program Menu", explained in the following pages.

- - - - -

If you are viewing a video or an image in full screen, these buttons are not visible.

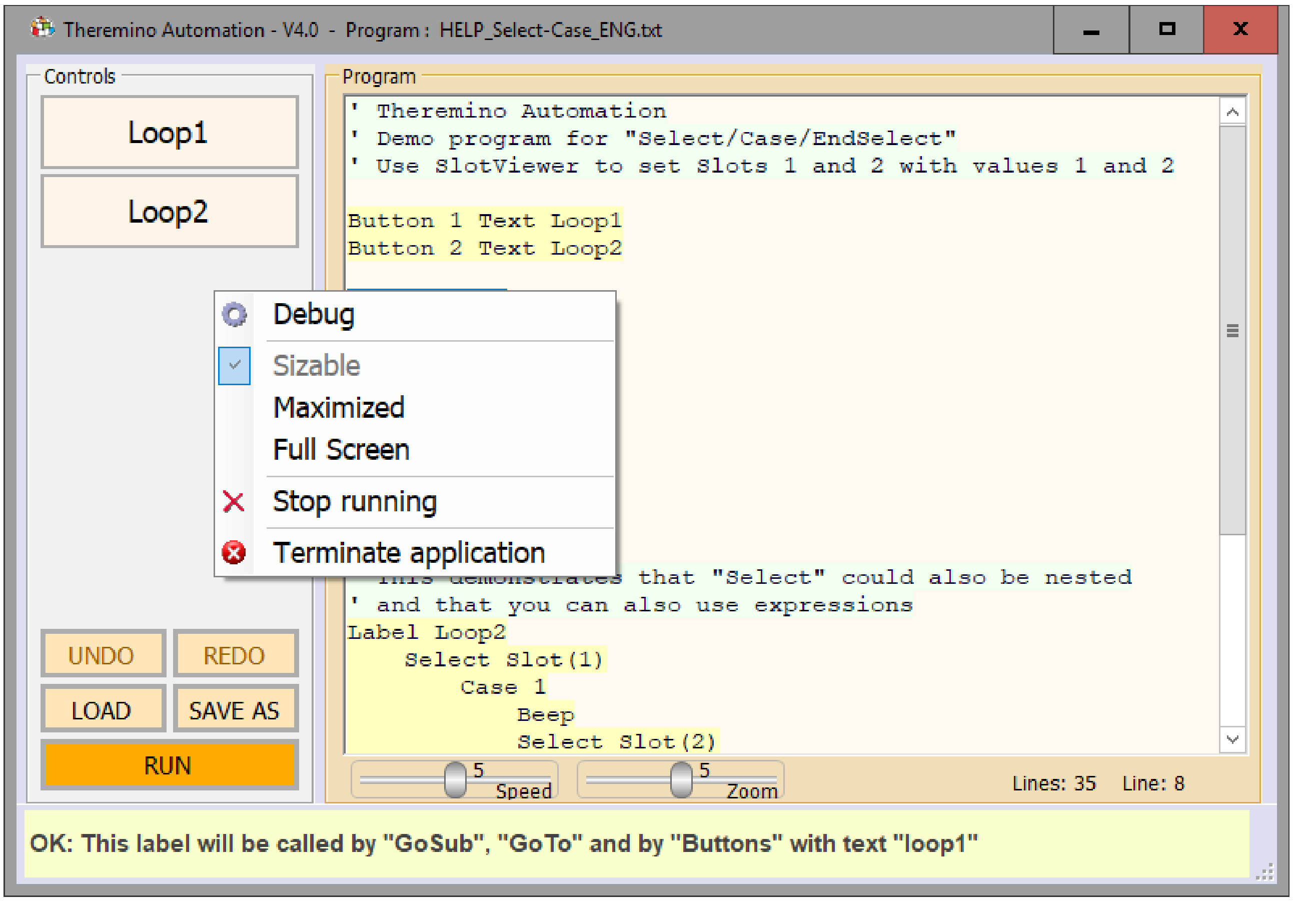
On those occasions, to stop the program, you can use the following methods:

- You can stop running the program at any time, with the key **SHIFT-ESC,** or with **ALT-E.**

- You can use the right mouse button and open the application menu (shown on the next page).

The application menu

By clicking the right mouse button (or by tapping the touch screen without lifting your finger for two seconds), you open this menu.



Theremino Automation - V4.0 - Program : HELP_Select-Case_ENG.txt

```
' Theremino Automation
' Demo program for "Select/Case/EndSelect"
' Use SlotViewer to set Slots 1 and 2 with values 1 and 2

Button 1 Text Loop1
Button 2 Text Loop2
```

```
' This demonstrates that "Select" could also be nested
' and that you can also use expressions
Label Loop2
    Select Slot(1)
        Case 1
            Beep
            Select Slot(2)
```

Lines: 35 Line: 8

OK: This label will be called by "GoSub", "GoTo" and by "Buttons" with text "loop1"

When the program is stopped, you must click on the left side of the application (control area), and not on the program area.

Instead, when the program is started, this menu appears when you click anywhere, even on the program, as well as on the pictures and videos to full screen.

This menu allows you to open the Debug window, choose the size of the window, stop the execution of the program, and also totally close the Automation application.

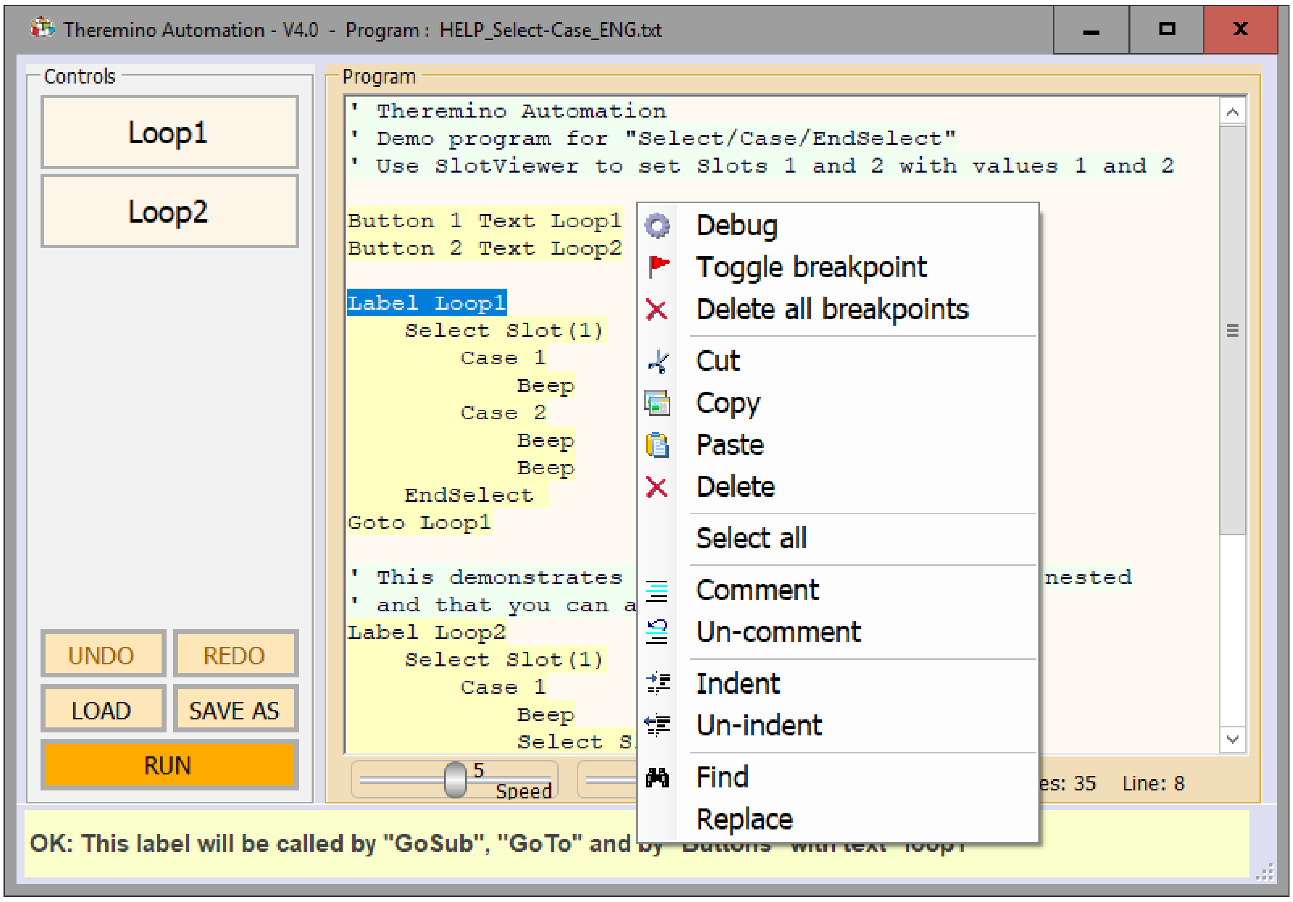
Knowing about this menu is important.

Without him never be able to get out of some situations.

For example, when a full-screen video is active.

The program menu

By clicking on the program area, with the right mouse button (or by tapping the touch screen without lifting your finger for two seconds), this menu opens.



```
' Theremino Automation
' Demo program for "Select/Case/EndSelect"
' Use SlotViewer to set Slots 1 and 2 with values 1 and 2

Button 1 Text Loop1
Button 2 Text Loop2

Label Loop1
    Select Slot(1)
        Case 1
            Beep
        Case 2
            Beep
            Beep
    EndSelect
Goto Loop1

' This demonstrates                              nested
' and that you can a
Label Loop2
    Select Slot(1)
        Case 1
            Beep
            Select S
```

es: 35   Line: 8

OK: This label will be called by "GoSub", "GoTo" and by Buttons with text "loop1"

The program should not be running, otherwise instead of this menu would open the one on the previous page.

The first three lines of this menu are used for Debug (software maintenance and monitoring) and will be better explained ***This Page***.

The commands **Cut**, **Copy**, **Paste**, **Delete** and **select All**, they copy, paste, delete and sort the selected parts of the program. In their place, you could also use CTRL-X, CTRL-C, CTRL-V, DELETE and CTRL-A.

With **Comment** and **Uncomment** you comment (add initial quote) to whole areas of the program. Or you delete comments.

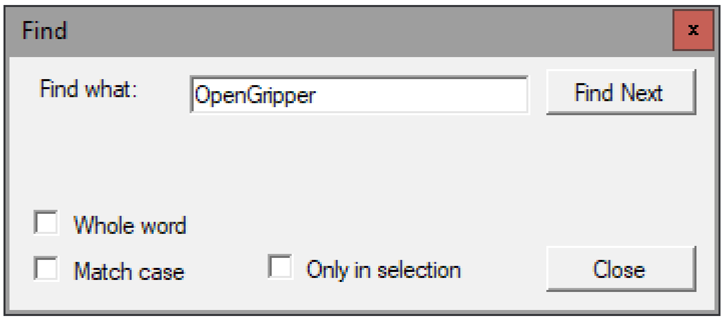
Indent and **Unindent** act only on commented lines.

With **Find** (or CTRL-F), and **Replace,** you open the windows to find and replace words and phrases.

Find and Replace

With the latest voices at the bottom of the application menu, you open two windows, similar to each other.

The "FIND" window

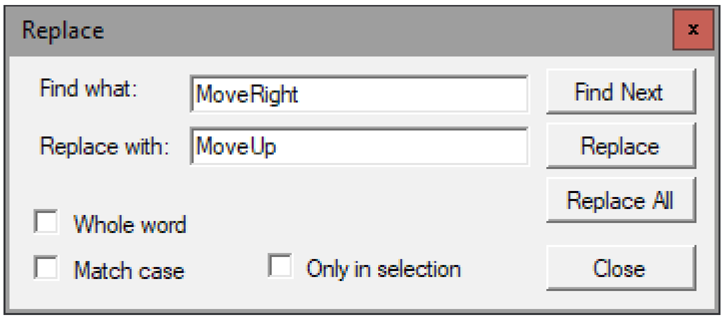


Find what: OpenGripper
☐ Whole word
☐ Match case
☐ Only in selection

With this window you look for words or phrases in the text of the program.

- If you enable "Whole Word", the word must be complete.
- If you enable "Match case", the word should also match as case sensitive.
- If you enable "Only in selection", the word is searched only in selected text.
- With "Find next" (or F3), you go to the next occurrence of the searched word. If you get to the end of the research program starts from the beginning.

The "REPLACE" window



Find what: MoveRight
Replace with: MoveUp
☐ Whole word
☐ Match case
☐ Only in selection

This window is the same as the previous, but also lets you replace the word (or phrase) with another.

If you press "Replace" is carried out only one replacement. But with "Replace All" you replace all occurrences in all the program or in the selection.

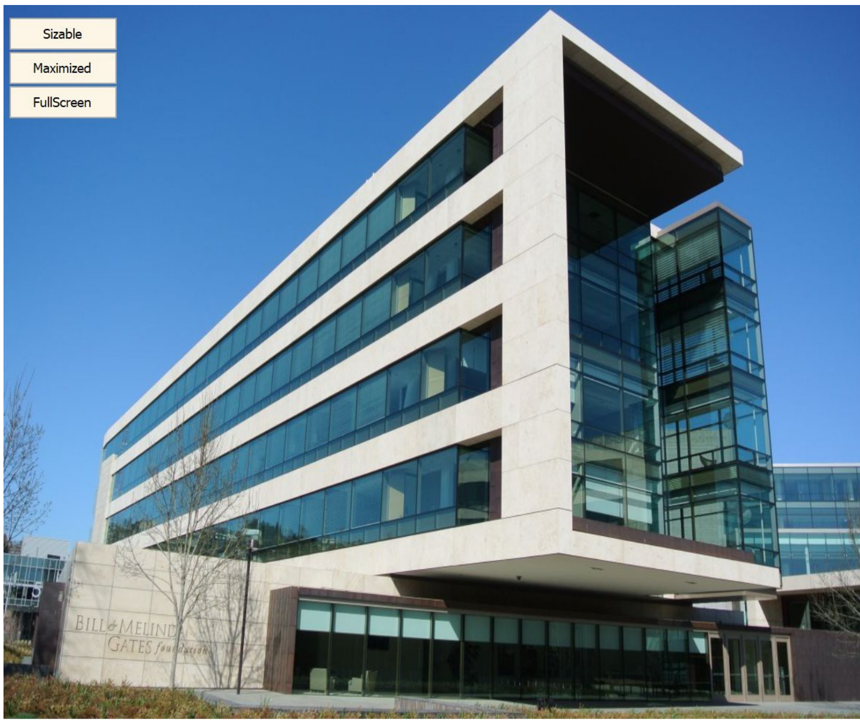
Full Screen operation

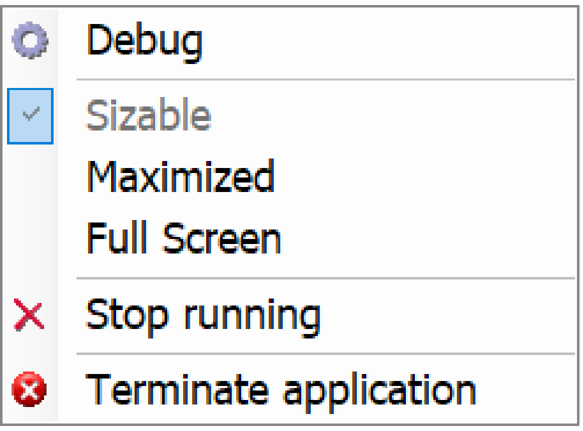
"Full screen" means that the display has no visible window edges, and that the desktop bars are hidden.

Automation can view in “Full screen” all the pictures and video.

Even web pages, and the text of the program, are displayed in full screen, but with the buttons vertical bar on the left.

If the program enables some buttons, then it will be visible overlapping pictures, and videos, as seen in this image.





To open the window in full screen, you can use the menu that opens, right-click the mouse, and choose "Full Screen".

To return to the “in window” operation, you choose the "Maximized" and "Sizable" options.

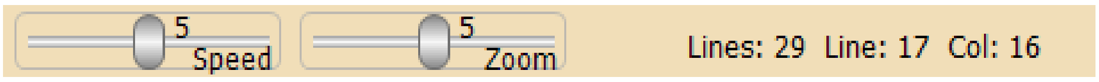
Alternatively, to exit from the FullScreen condition, you might interrupt the execution of the program, with the buttons **SHIFT-ESC** or **ALT-E.**

While the program is running you can also use the **`Window Fullscreen`**, **`Window Maximized`** and **`Window Sizable`** instructions which are explained in ***this page***.

- - - - - -

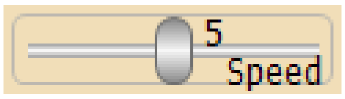
To experiment with the "Windows" instruction,
and with the various dimensions of the window,
see the "Demo – Windows" example.

The bottom bar



Lines: 29 Line: 17 Col: 16

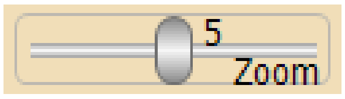
In this bar are the controls for: speed, size, lines and columns of the program.



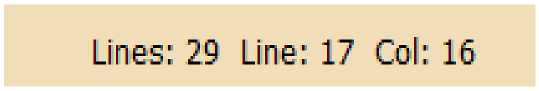
The SPEED slider sets the program execution speed. The speeds range from "1" (one instruction per second), until "8" (ten thousand instructions per second), and to "9" (the maximum speed allowed by the system).

While writing the program it is good to use an average speed. Usually the speed "5" (20 instructions per second), Which is slow enough to be able to visually follow the execution of the program.

In some cases you could write the statement ***Speed*** in the same program, and in this case the Speed slider would no longer be considered.



The ZOOM slider sets the size of text, both in the program window, or in the Debug window.



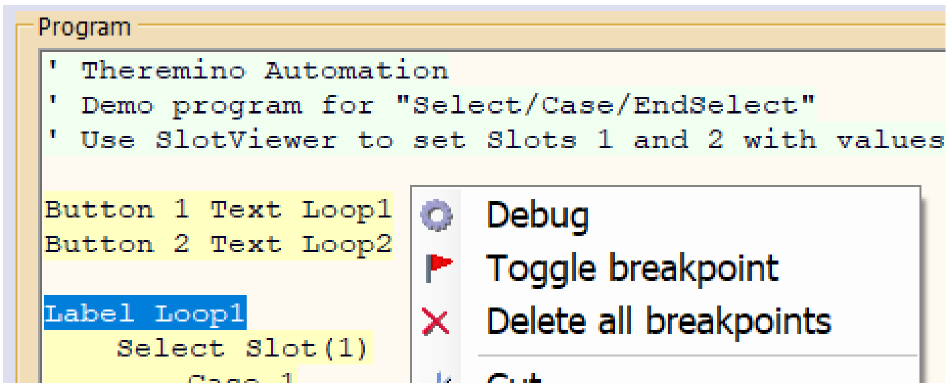
Lines: 29 Line: 17 Col: 16

The right side of the lower bar shows information about the program:

- The total number of lines
- The line where the cursor is (starting from line 1)
- The column where the cursor is (starting from column 1)

The Debug Window

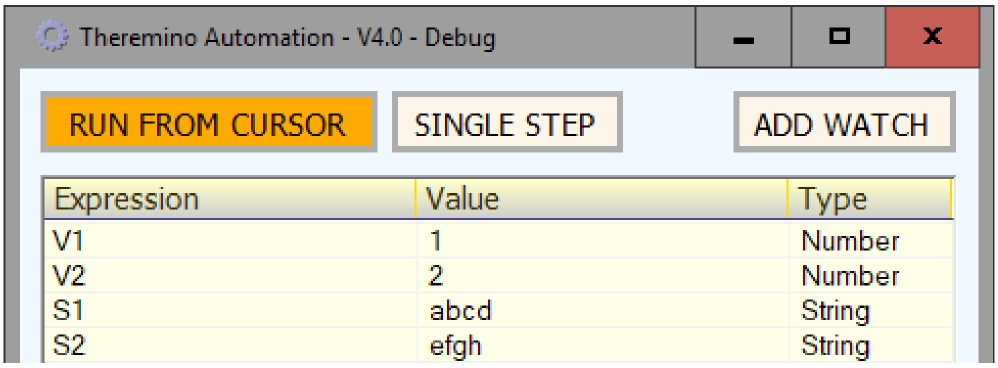
The "Debug" window facilitates the development of the software. To open it you press the right button of the mouse on the map area, and choose "Debug".



```
Program
' Theremino Automation
' Demo program for "Select/Case/EndSelect"
' Use SlotViewer to set Slots 1 and 2 with values

Button 1 Text Loop1
Button 2 Text Loop2

Label Loop1
    Select Slot(1)
```



Expression	Value	Type
V1	1	Number
V2	2	Number
S1	abcd	String
S2	efgh	String

The two main functions are: "RUN FROM CURSOR", which allows you to run the program from anywhere, and "SINGLE STEP", which allows you to run one line at a time.

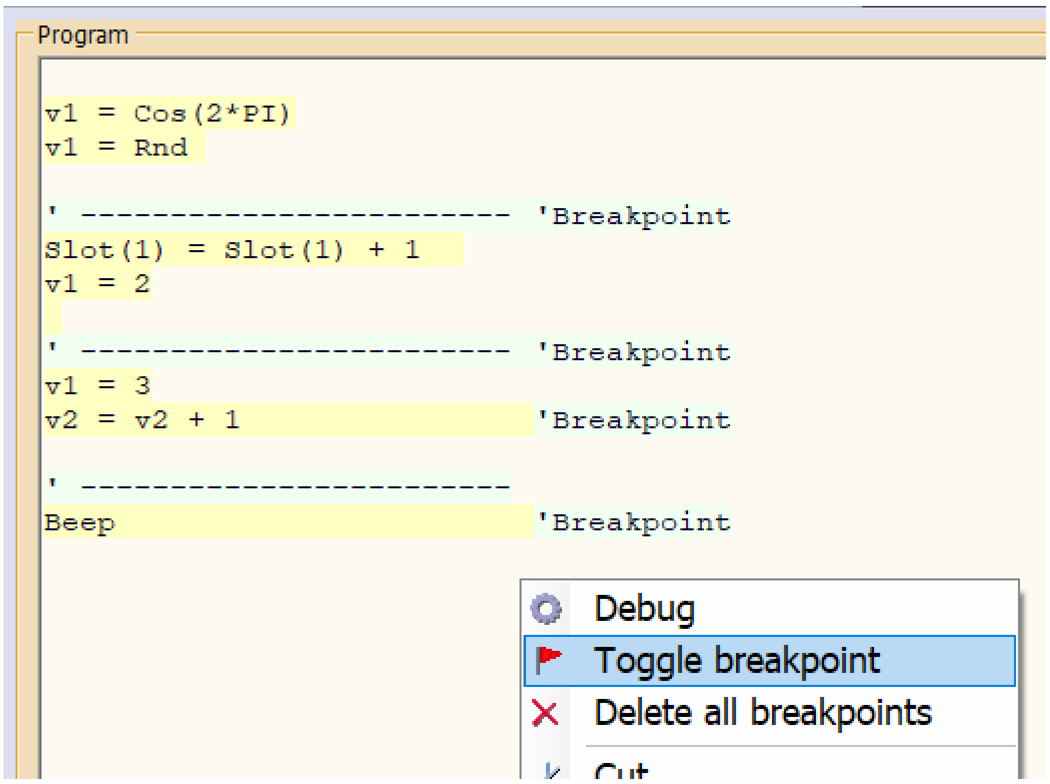
To run the program from an arbitrary point, first of all the stops with the STOP button in the main window. Alternatively, you can stop it by pressing RUN FROM CURSOR or SINGLE STEP. Then you place the cursor on the line you want to execute, and you press one of these buttons.

The other functions of this window are: The Watch (watch expressions), the exec line (execute instructions) and BreakPoints, which will be explained in the following pages.

- - - - -

Experiment with the examples of the "Demo Debug" folder

The Debug Window (Breakpoints)



```
v1 = Cos(2*PI)
v1 = Rnd

' ------------------------   'Breakpoint
Slot(1) = Slot(1) + 1
v1 = 2

' ------------------------   'Breakpoint
v1 = 3
v2 = v2 + 1                  'Breakpoint

' ------------------------
Beep                         'Breakpoint
```

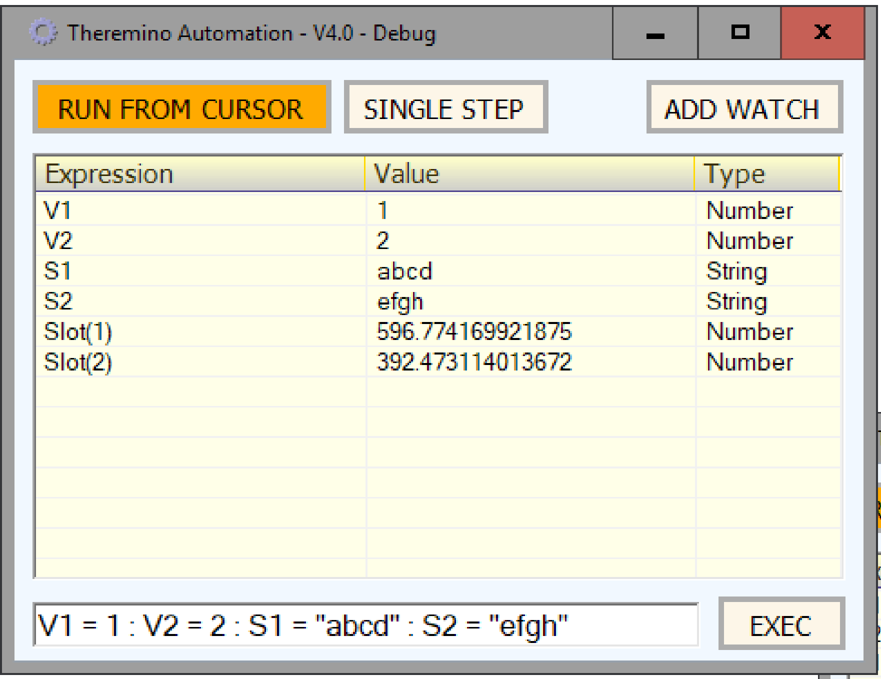
When the Debug window is open, the program stops at every line ending with `'Breakpoint` .

Instead, If the Debug window is closed, Breakpoints are ignored. So it is possible to leave them in the preferred positions, even in the final program.

When the program is stopped at a breakpoint, you can use all the options in the Debug window.

- You can explore the values of the variables (from v1 to v9 and s1 to s9), and the slot (0 to 999), with the watch table.
- In the Watch you can also write complex expressions and calculations.
- You can edit the values of variables and Slot, writing assignments in the EXEC line
- You can execute instructions, writing in the low line and pressing EXEC
- You can continue to run with RUN FROM CURSOR
- You can run a single line, with SINGLE STEP
- You can change the position of running, and then continue with RUN FROM CURSOR or SINGLE STEP

The Debug Window (Watches)

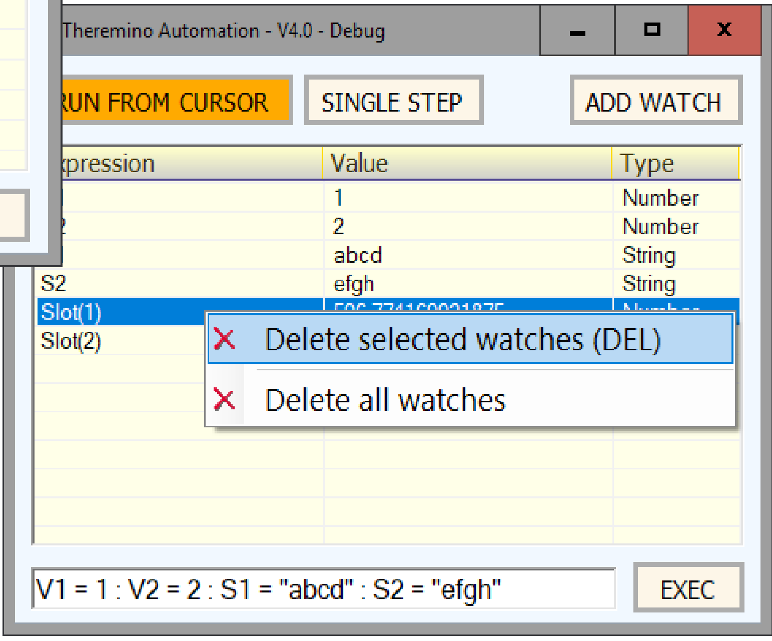


Expression	Value	Type
V1	1	Number
V2	2	Number
S1	abcd	String
S2	efgh	String
Slot(1)	596.774169921875	Number
Slot(2)	392.473114013672	Number

```
V1 = 1 : V2 = 2 : S1 = "abcd" : S2 = "efgh"
```

Each row of this table is a "Watch" (control expression).

With the watch you look at the values of variables and their type, and controls the operation of the program expressions.


Theremino Automation - V4.0 - Debug
RUN FROM CURSOR
SINGLE STEP
ADD WATCH
Delete selected watches (DEL)
Delete all watches
EXEC

The Watch are added by selecting a variable, function, or expression of program, and then pressing the "ADD WATCH" button.

To delete one or more watch, you select the chosen lines, pressing the right mouse button on the table, and you use the menu visible on the right.

To edit a watch you double-click with the left mouse button, on his line, and change the text of the expression with the keyboard.

The Automation application Watches are very powerful, you can write complex expressions, make calculations, check if expressions are true or false, read the values of the slot, add text strings, etc...

Moreover, unlike almost all programming environments, The Watch are updated in real time (ten times per second), even with the program stopped. So they always reflect the actual value of the expressions and Slots. Not even DotNet does this, and it is a very useful feature.

- - - - -

Experiment with the examples of the "Demo Debug" folder

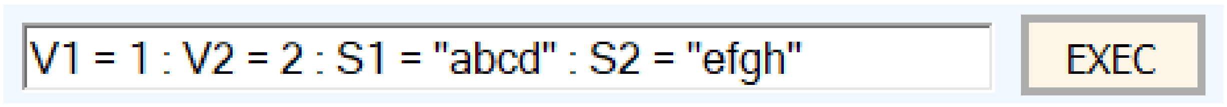
The Debug Window (Exec)

The bottom line of the Debug window, you can execute instructions and assign values to variables and to Slot.

Using the word Print, you can print complex expressions results, and then have them calculate and know the result.

You can also write the Goto and Gosub, and then press EXEC and jump to the corresponding labels, even while the program is running.

The line also accepts multiple instructions on the same line (separated by a colon, as shown in the example below).



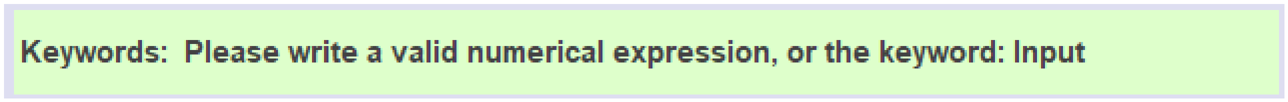
```
V1 = 1 : V2 = 2 : S1 = "abcd" : S2 = "efgh"
```

You can use just about any instructions that may be written in the program.

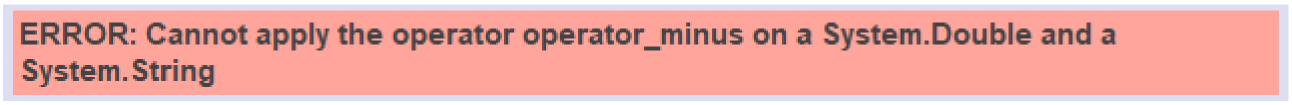
The only keywords not valid are: If, Else, Endif, Select, Case, CaseElse, EndSelect, Return, Stop and Wait. Writing them do not get any effect, they just do not run.

- - - - -

When you write in the EXEC line, the errors and suggestions are shown in the low bar of the main window.



Keywords: Please write a valid numerical expression, or the keyword: Input



ERROR: Cannot apply the operator operator_minus on a System.Double and a System.String

More info about errors and suggestions in the page "***Keywords***".

- - - - -

Experiment with the examples of the folder "Demo Debug"