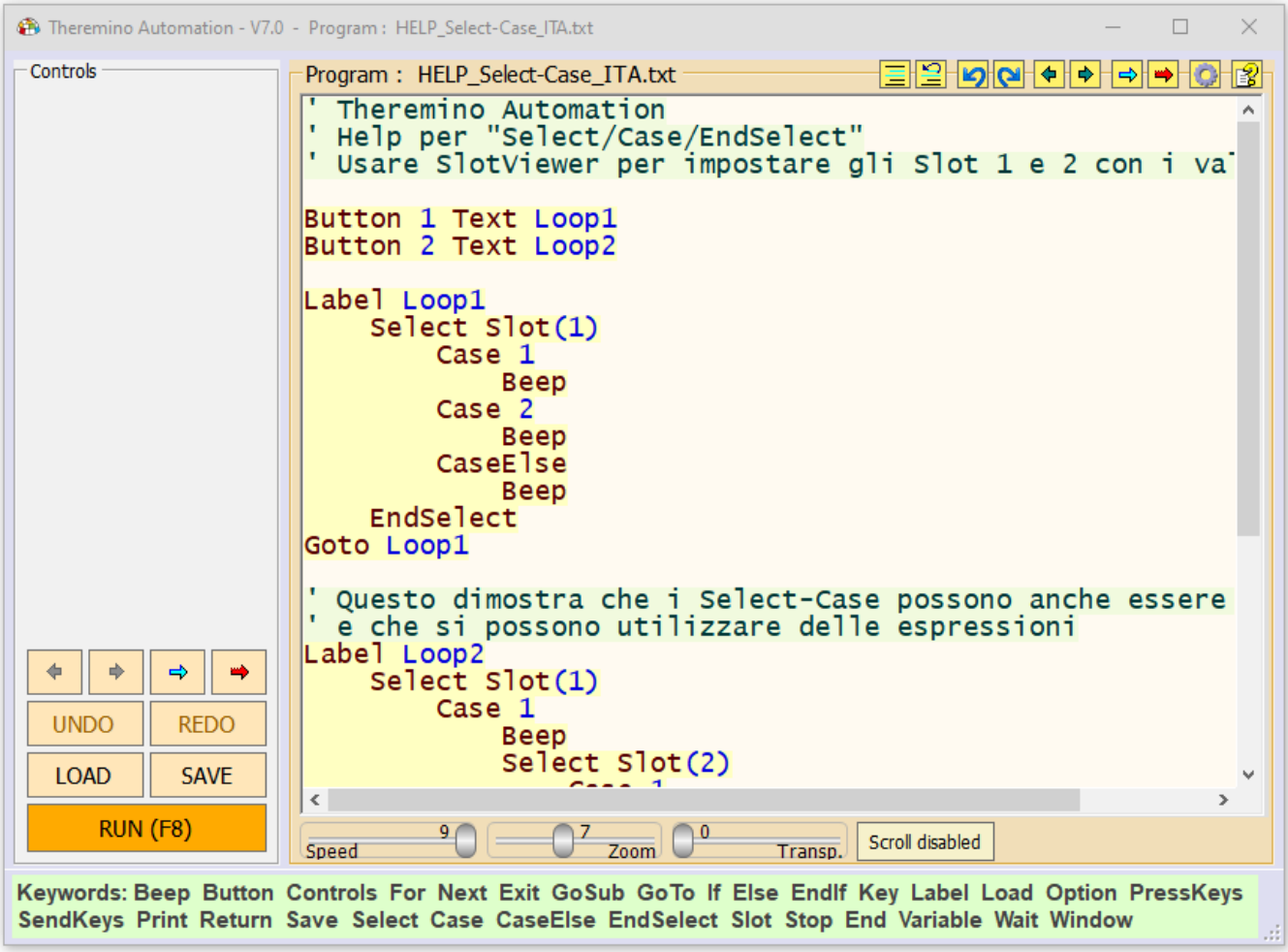
Sistema theremino



Theremino Automation V7.9

Indice generale

Premesse

Abbiamo semplificato e minimizzato tutto il possibile, per facilitare chi non ha esperienza nella programmazione.

Questo linguaggio è estremamente facile da usare e da capire, ma sono anche possibili operazioni complesse, come eseguire audio e video, o navigare in internet.

Con una sola istruzione si eseguono operazioni che in altri linguaggi richiederebbero conoscenze specialistiche e molte pagine di codice.

Limiti di Theremino Automation.

Keywords: Beep Button Controls For Next Exit GoSub GoTo If Else EndIf Key Label Load Option PressKeys SendKeys Print Return Save Select Case CaseElse EndSelect Slot Stop End Variable Wait Window

In Automation le istruzioni sono poche, una ventina. Non esistono le classi i tipi, le strutture, i thread e nessuno dei meccanismi complessi dei classici linguaggi di programmazione.

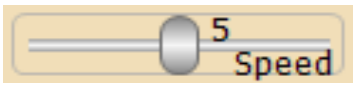
Quindi, per compiti complessi, si dovrebbe passare a ambienti di programmazione più potenti (ad esempio Visual Studio). Come linea guida si possono contare le righe, oltre le mille o duemila righe è meglio passare a un linguaggio più potente.

Theremino Automation è un linguaggio interpretato.

Nei linguaggi “interpretati”, a differenza di quelli “compilati”, le istruzioni non sono precompilate, ma vengono lette, carattere per carattere, e interpretate, durante l'esecuzione stessa.

L'esecuzione di un linguaggio interpretato è quindi più lenta. Nel nostro caso le istruzioni sono mediamente dieci volte più lente, rispetto alle stesse istruzioni scritte in Visual Studio (Csharp, C++ o VbNet).

La velocità è comunque sovrabbondante per i semplici compiti a cui è destinato questo linguaggio. Infatti abbiamo dovuto aggiungere la possibilità di rallentarlo, anche di migliaia di volte, per facilitare la comprensione di quello che accade.



Eseguire più istanze di Automation

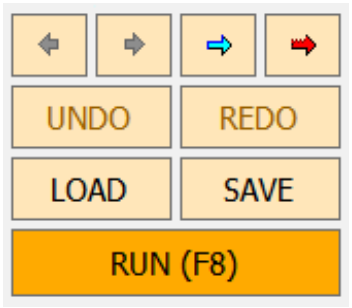
Se provate a lanciare una seconda istanza di Automation dallo stesso file questa non si aprirà. Si tratta di un comportamento voluto per impedire di aprirne due per sbaglio e anche per evitare che le opzioni delle due istanze si mischino. Se necessario si può forzare l'apertura di una seconda istanza del programma tenendo premuto SHIFT durante l'avvio.

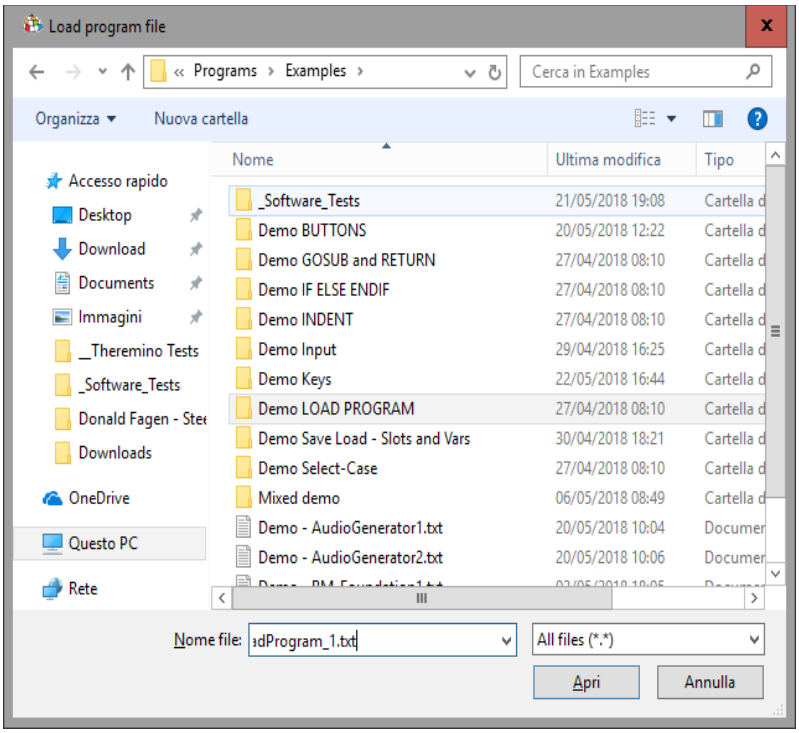
Se si vogliono due o più copie di Automation che possano funzionare contemporaneamente, e ciascuna con opzioni indipendenti, basta copiare il file "Theremino_Automation.exe" con un diverso nome.

Se si preferisce si possono tenere i file "Theremino_Automation.exe" in cartelle separate o anche tutti nella stessa cartella.

Caricare e eseguire i programmi

Utilizzando il pulsante LOAD si apre una finestra che permette di scegliere i programmi da caricare.





Dopo aver caricato il programma lo si esegue con il pulsante RUN.

Per fermare il programma si preme il tasto RUN (che è diventato STOP),

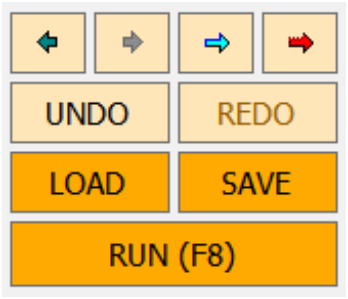
Si può anche fermare il programma facendo click sul testo del programma.

Modificare e salvare i programmi

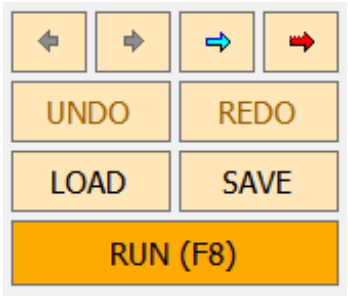
Ogni modifica che si fa a un programma viene automaticamente salvata, per cui non c'è bisogno di ricordarsi di salvarlo, prima di chiudere la applicazione Automation, o di caricare un altro programma.

Quando si riaprirà il programma sarà esattamente come lo si era lasciato.

Quindi, se non si vuole perdere l'originale, prima di modificare un programma è bene salvarlo con un diverso nome, utilizzando il tasto SAVE.

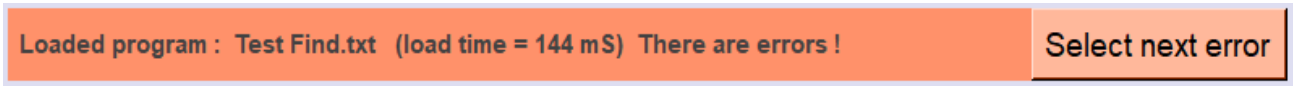


I tasti LOAD e SAVE evidenziati in arancione indicano che il programma è stato modificato e non ancora salvato.



- **Premendo SAVE con il pulsante destro del mouse** si salva velocemente il programma.

- **Premendo LOAD con il pulsante destro del mouse** si salva il programma e poi lo si ricarica immediatamente. Con questa operazione il programma viene controllato dall'inizio alla fine e se contiene errori la riga inferiore della applicazione mostrerà un messaggio.



Se il programma contiene errori allora nella riga inferiore appare il pulsante "Select next error", cliccandolo si scorrono tutte le righe con errori.

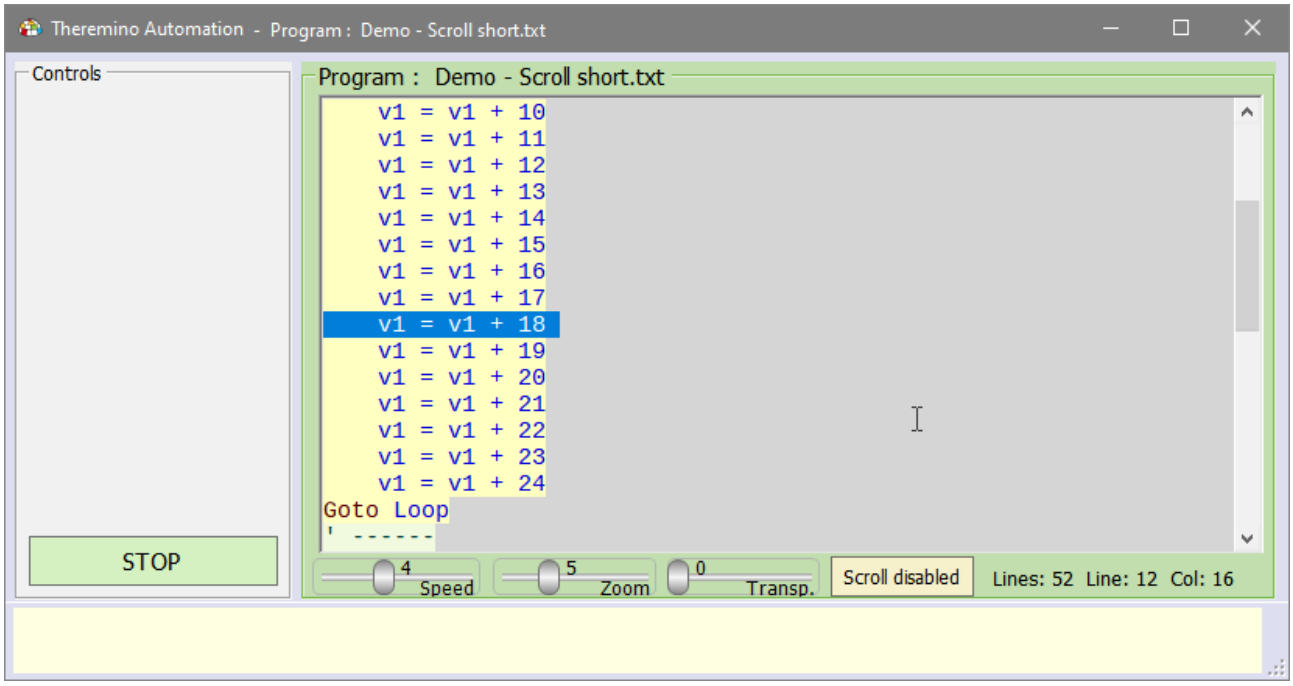
- - - - -

E' anche possibile inviare tutte le vecchie versioni dei programmi nel cestino del sistema, vedere l'opzione "Recycle bin" in ***questa pagina***.

In caso di bisogno, si potranno anche recuperare gli esempi originali dal file ZIP originale, che si è scaricato dal ***sito theremino***.

Esecuzione del programma

Durante l'esecuzione del programma la riga in esecuzione viene evidenziata con il fondo blu, come nella immagine seguente. Nelle nuove versioni di Automation (dalla 7.0 in poi) vengono anche evidenziate le ultime righe eseguite, come si vede nella prossima pagina.



La finestra del programma si comporta in modi diversi a seconda che il pulsante "Scroll" sia abilitato o no.

Se lo scroll è abilitato ("Scroll enabled")

- La finestra viene fatta scorrere in verticale per mostrare sempre la linea eseguita.
- Se la velocità "Speed" è bassa (da 1 a 4) allora la linea evidenziata viene anche mantenuta al centro dello schermo.

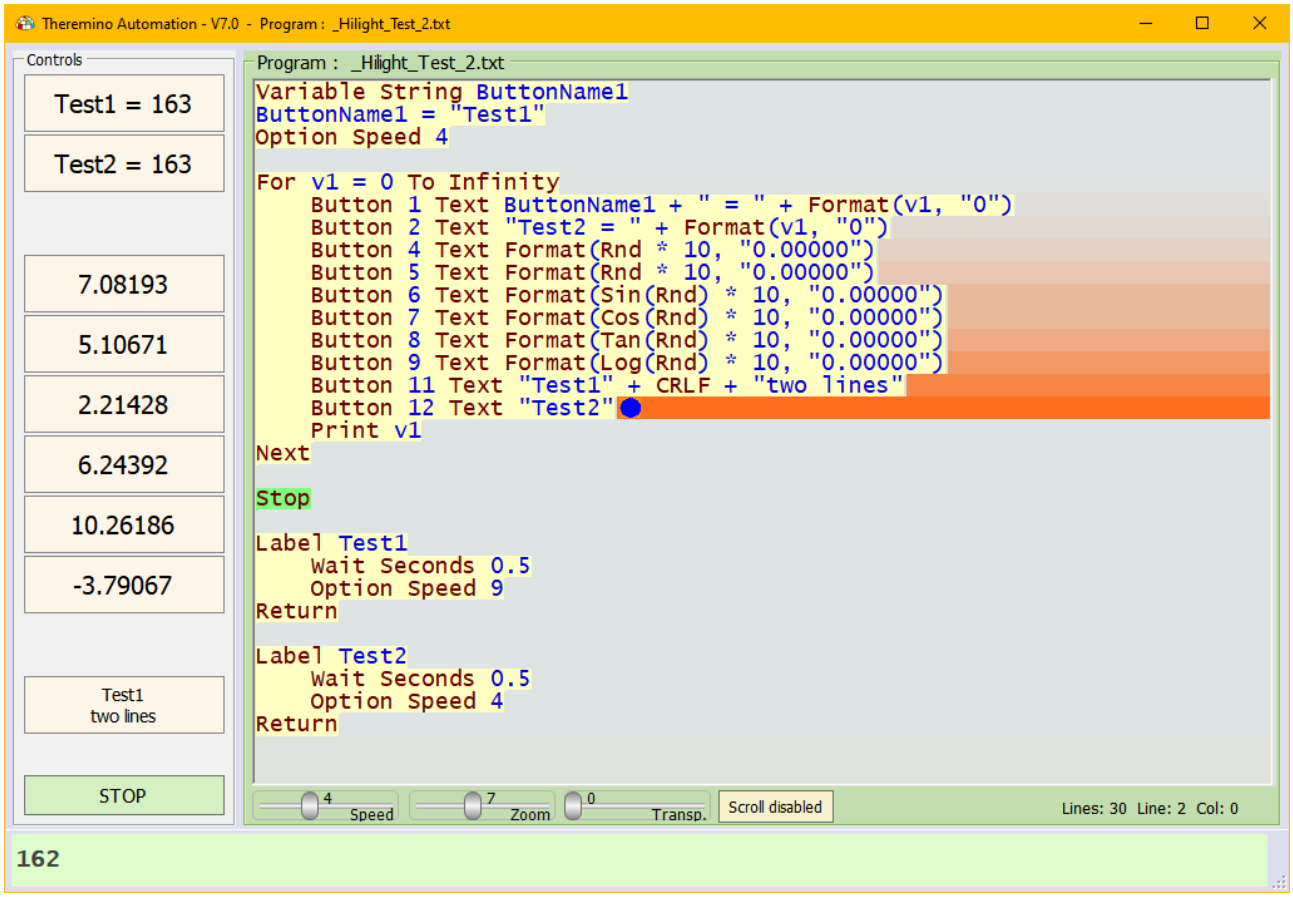
Se lo scroll è disabilitato ("Scroll disabled")

- Le linee in esecuzione vengono evidenziate, ma la finestra non scorre per mostrarle.
- Si può quindi utilizzare la rotella del mouse, o la barra di scorrimento verticale, per vedere stabilmente la zona preferita del programma.

Provate anche gli esempi della cartella
"Demo programs\Demo Execution and Scroll"

Visualizzazione delle righe eseguite

Dalla versione 7.0 in poi le linee eseguite vengono evidenziate in colore rosso, come visibile in questa immagine.



Il cursore blu indica la linea eseguita istantaneamente e viene seguito da una colorazione rossa dello sfondo. Questa colorazione si attenua lentamente fino a ripristinare il colore grigio chiaro dello sfondo dopo circa un secondo.

Se il programma viene eseguito lentamente, ad esempio a velocità 4, allora si vedranno le ultime righe eseguite con gradazioni differenti.

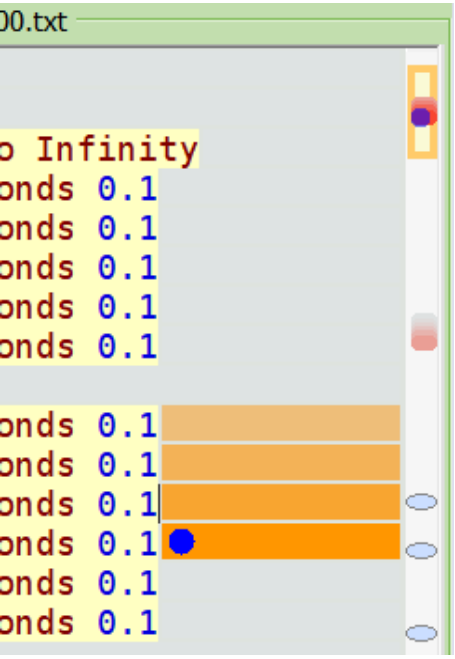
Se il programma viene eseguito velocemente allora si vedranno blocchi di linee colorarsi insieme e attenuarsi tutte insieme.

Queste colorazioni sono molto utili per seguire l'andamento del programma. Si consiglia di mantenere il pulsante Scroll disabilitato e di scorrere manualmente il programma con la rotella del mouse o con il cursore della finestra e anche di visualizzare parti del programma, cliccando i Button con il tasto CTRL premuto come spiegato alla fine della prima pagina sui "Button".

Provate anche gli esempi della cartella
"Demo programs\Demo Hilight Running Lines"

La barra verticale e i segnalibri

La barra verticale ha due comportamenti diversi a seconda che il programma sia in esecuzione o in editing.



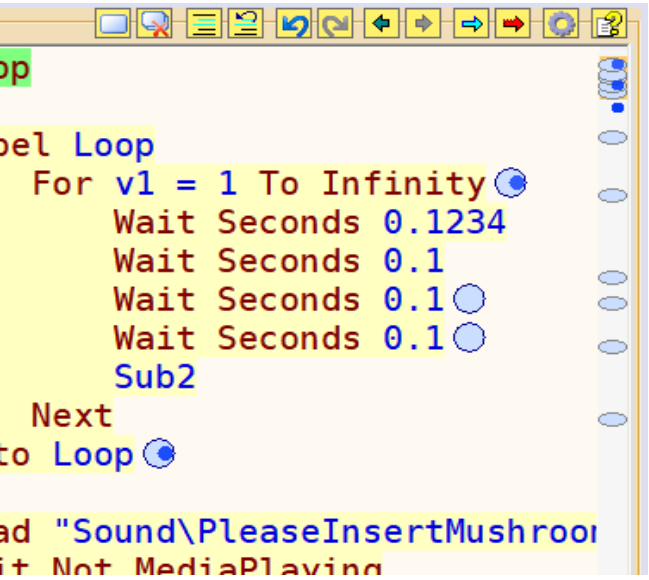
Durante l'esecuzione del programma nella barra verticale appaiono tacche di colore arancione in movimento.

Portando il cursore sopra alle parti colorate della barra la finestra principale mostra le parti di programma in esecuzione.

Le tacche arancioni sono di colore degradante, quelle di colore più intenso si riferiscono alle ultime righe eseguite.

Il pallino blu indica la riga di programma che è attualmente in esecuzione.

Se il programma è fermo in attesa di qualcosa può accadere che le bande arancioni sfumino completamente nel colore grigio di sfondo e in tal caso il pallino blu serve per individuare dove si è fermata l'esecuzione.

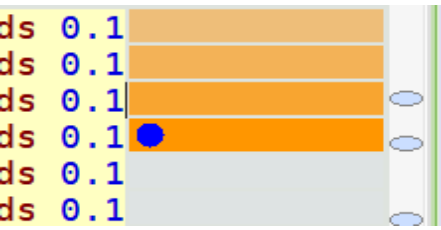


Quando il programma non è in esecuzione nella barra verticale appaiono informazioni utili per individuare le varie parti del programma.

I cerchietti azzurri sul programma e gli ovali azzurri nella barra indicano le righe con i segnalibri.

I cerchietti e le tacche blu indicano le ultime righe del programma che sono state modificate.

Le righe e le tacche verdi indicano tutte le occorrenze della parte di programma che è stata selezionata (in questo caso la parola "Loop")

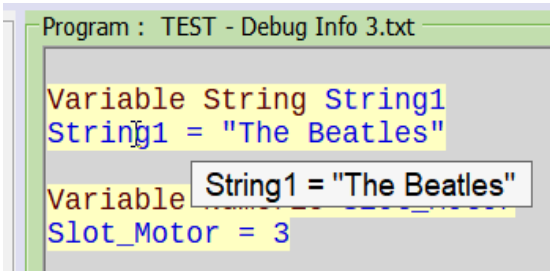


I segnalibri (ovali e cerchietti azzurri) sono visibili anche durante l'esecuzione del programma e sono molto utili per marcare zone importanti e trovarle velocemente.

Controllare i valori durante l'esecuzione

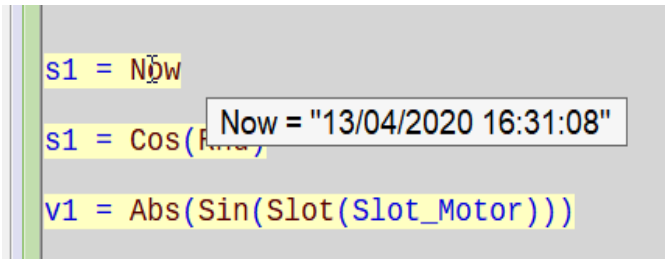
Nelle pagine precedenti abbiamo visto che si può fermare lo scorrimento automatico, quindi con la rotella del Mouse si possono ispezionare varie parti del programma.

Volendo si può anche usare un ***tasto esterno*** per mettere in pausa la applicazione, oppure si potrebbero posizionare istruzioni Stop in punti strategici, o inserire dei Breakpoint e aprire il pannello Debug. Quando il programma è in funzione, o il pannello Debug è aperto, si possono conoscere i valori di variabili e funzioni spostando il cursore del Mouse su di esse.

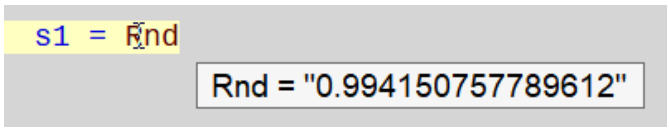


Posizionando il mouse su una variabile si può conoscere immediatamente il suo valore, come mostrato in questa immagine.

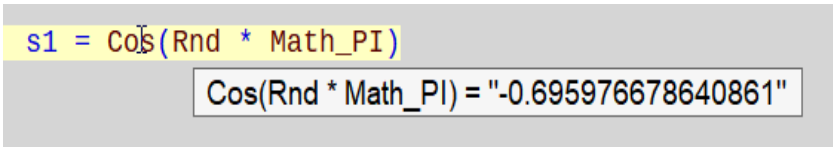
Notare che il cursore del Mouse (poco visibile) è posizionato tra la "n" e la "g" di "String1"



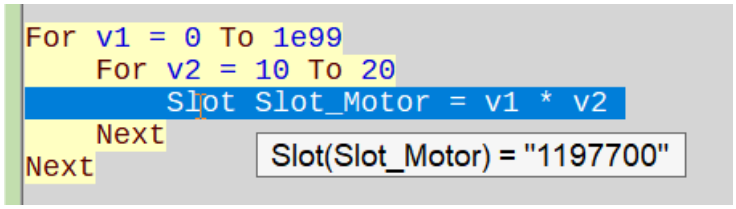
Si possono interrogare anche le funzioni. Ad esempio la funzione "Now" risponde un valore che cambia nel tempo ogni secondo.



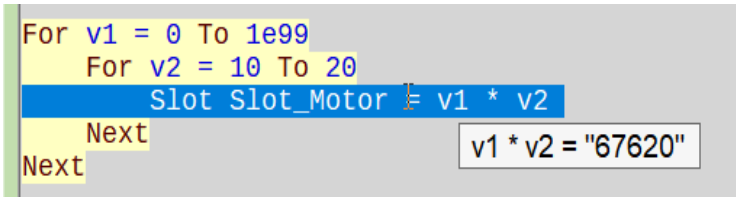
Anche la funzione Rnd viene continuamente ricalcolata e cambia nel tempo.



Posizionando il cursore su "Cos" si ottiene il risultato di una funzione complessa.



Si possono anche conoscere istantaneamente i valori numerici degli Slot.



E anche i risultati di formule numeriche (per definire parti interessanti le si possono racchiudere tra parentesi)

Sperimentare con gli esempi della cartella
""Demo programs\Examples \ Demo FastDebug"

Semplici programmi per iniziare

Chi programma per la prima volta potrebbe trovare utili i semplici esempi che si trovano nella cartella “Simple Programs”.

Il modo migliore per iniziare è caricare questi esempi, ed eseguire le istruzioni una per volta, con il pulsante “Single Step” della finestra di “Debug” (per aprirla utilizzare il tasto destro del Mouse e scegliere Debug).

Nella finestra di Debug sono già presenti alcune righe che mostrano il valore delle variabili, degli Slot e delle espressioni. Ma è anche facile aggiungerne di nuove.

Premendo ripetutamente Single Step, si possono vedere i valori cambiare e capire come procede il programma, e come vengono eseguiti i calcoli e assegnati i valori.

Ecco un esempio di semplice programma

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

Da notare che, dopo aver eseguito l'ultima riga, il programma riparte automaticamente dalla prima riga. Nel caso si volesse fermarlo si dovrebbe aggiungere una riga “Stop” alla fine.

Ecco un secondo esempio

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

In questo esempio l'istruzione Rnd assegna, alla variabile v1, un valore casuale tra 0 e 1. Poi emette un suono se è maggiore di 0.5, altrimenti stampa il suo valore.

- - - - -

Vedere anche gli altri esempi che si trovano nella cartella “Simple Programs”

Struttura dei programmi

I programmi più semplici hanno una prima sezione di "inizializzazione" che termina con uno STOP e di seguito tutte le procedure che iniziano con LABEL e che verranno eseguite solo quando verranno chiamate da un pulsante o altri eventi.

```
Button 1 Text "Button1"
Button Button1 Color Yellow7

Stop

Label Button1
     Beep
Return
```

Programmi più complessi, che prevedono azioni da eseguire ripetutamente, prima dello STOP hanno un ciclo chiamato MainLoop che viene eseguito di continuo.

```
Button 1 Text "Button1"
Button Button1 Color Yellow7

Label MainLoop
     Gosub CheckMotorTemperature
     ' ------------------------------------------------------
     Wait Seconds 0.1 ' <<< 0.02 sec. or more
     ' ------------------------------------------------------
Goto MainLoop

Stop

Label Button1
     Beep
Return

Label CheckMotorTemperature
     ...
     ...
Return
```

Il ciclo MainLoop deve contenere una istruzione Wait Seconds 0.02 (0.02 è il tempo minimo da utilizzare) altrimenti viene eseguita così velocemente da non lasciare più tempo per altre operazioni come la esecuzione dei tasti della tastiera o dei pulsanti.

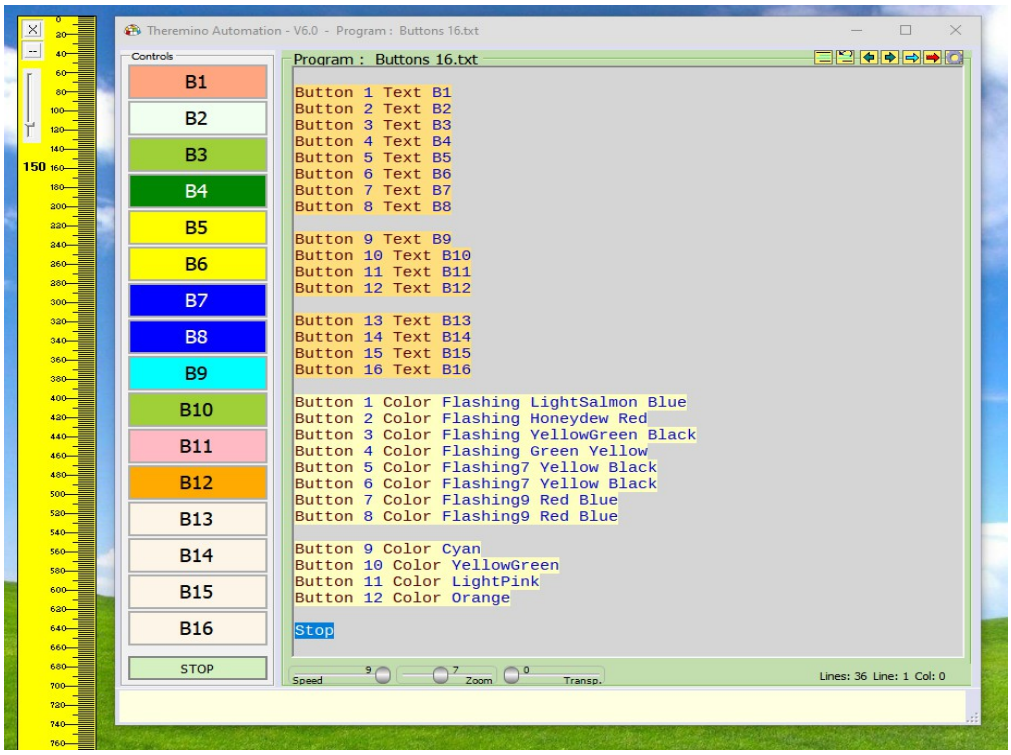
Nel ciclo MainLoop non si devono invece utilizzare pause troppo lunghe (oltre i 100 millisecondi) e non si devono chiamare procedure lente, altrimenti il ciclo non viene ripetuto abbastanza velocemente e le operazioni che esegue vengono rallentate.

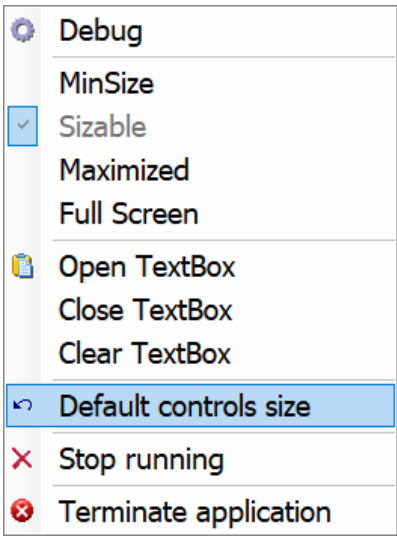
Gli eventi (Button, Keys e altri) interrompono il MainLoop e quindi anche essi vanno eseguiti nel più breve tempo possibile.

Ridimensionare i controlli

Ridimensionando i controlli della applicazione, si possono visualizzare fino a 16 o 32 pulsanti in verticale anche su schermi da 1280 x 768.

In altri casi si potrebbero allargare i pulsanti in orizzontale per farvi stare più caratteri, oppure si potrebbero allargare o stringere tutti i controlli per ottenere una buona leggibilità a seconda della risoluzione dello schermo usato.





Per ridimensionare i controlli:

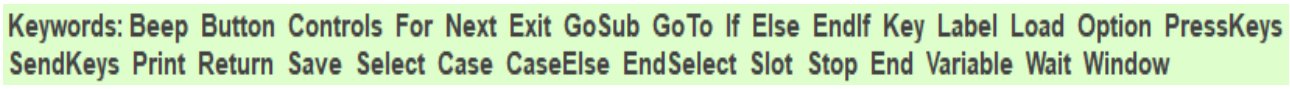
- Si posiziona il cursore dovunque in mezzo allo schermo.
- Poi si premono e si mantengono premuti i tasti CTRL e SHIFT
- E infine, senza premere i pulsanti del mouse, si sposta il Mouse nelle quattro direzioni.
- Oppure si premono le quattro frecce della tastiera per ingrandire o ridurre i controlli in verticale o in orizzontale.

Non potendo utilizzare CTRL e SHIFT (ad esempio su un tablet senza tastiera), consigliamo di chiudere la applicazione Automation e di modificare i valori "ControlsRatio_W" e "ControlsRatio_H" nel file "Theremino_Automation_INI.txt".

Per ripristinare le dimensioni originali dei controlli si apre il menu della applicazione e si fa click su "Default controls size".

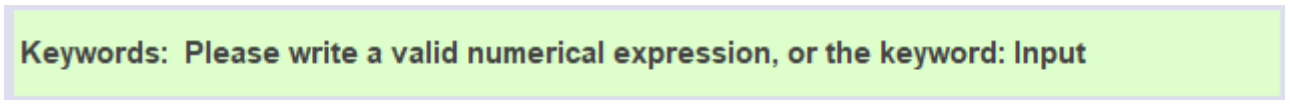
Parole chiave

Le parole chiave (keywords), sono una ventina in tutto, facili da ricordare e facili da usare.

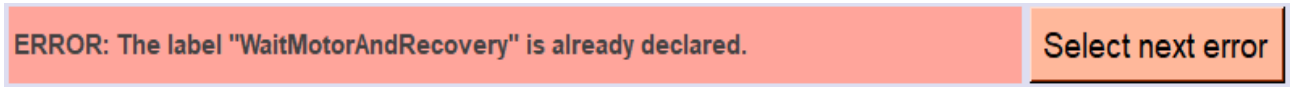


La barra inferiore della applicazione mostra le parole chiave utilizzabili.

Si possono scrivere velocemente le istruzioni, facendo Click con il mouse sulle parole chiave della barra inferiore.



Mentre si scrive, la barra inferiore mostra suggerimenti per completare le istruzioni correttamente.



Selezionando linee che contengono errori appare il pulsante "Select next error", e cliccandolo si evidenziano tutte le righe con errori, una dopo l'altra.

- - - - -

Purtroppo non tutti gli errori vengono individuati e spiegati, non siamo Microsoft, siamo in pochi e il nostro tempo è limitato. Comunque la barra inferiore è un buon aiuto per chi inizia, e ad ogni nuova versione funziona un po' meglio.

- - - - -

Nelle prossime pagine le parole chiave saranno spiegate nei dettagli.

Notare che sono elencate in ordine “quasi” alfabetico (alcune parole sono raggruppate perché simili tra loro).

Le parole chiave una per una

Array

Gli Array sono disponibili a partire dalla versione 7.8 di Automation e semplificano operazioni che nelle versioni precedenti sarebbero state difficili da implementare.

Si tratta di array numerici, con nomi predefiniti, da Array1 a Array9.

A parte queste limitazioni le potenzialità di questi Array sono notevoli.

- Ogni Array può memorizzare fino a dieci milioni di valori.
- Si possono utilizzare numeri interi molto grandi (da -9007199254740991 a 9007199254740991) senza perdita di precisione.
- Si possono utilizzare numeri in virgola mobile "Double precision" che sono ancora più grandi (da -1.7 * 10^308 a +1.7 * 10^308).

Inoltre il dimensionamento degli Array è automatico e non si producono mai errori anche se l'indice è minore di zero o maggiore della dimensione dell'array.

Ad esempio con la riga seguente l'Array2 viene automaticamente ridimensionato a 1000 posti (da 0 a 999) e il posto 999 viene aggiornato con il valore 1234.

```
Array2(999) = 1234
```

Se l'Array2 conteneva 1000 posti o più, allora la sua dimensione non cambia.

Durante il ridimensionamento gli elementi già presenti mantengono il loro valore.

ArrayClear

La dimensione degli Array può solo aumentare automaticamente, ma se necessario si può svuotarli. Ad esempio per azzerare Array1 si scrive: `ArrayClear(1)`

Oppure si può inizializzarlo scrivendo una serie di valori separati da virgole o spazi.

`Array1 = 111, 222, 333,444,555` o anche `Array1 = 111, 222 333 444 , 555`

questo caso è il numero di elementi a determinare la dimensione dell'Array.

Funzione ArrayLength

Questa funzione fornisce il numero di elementi di un Array. Ad esempio per stampare quanti elementi contiene l'Array1 si scrive: `Print ArrayLength(1)`

Funzione ArrayToString

Questa funzione fornisce una stringa che contiene tutti gli elementi di un Array separati da spazi. Ad esempio `Print ArrayToString(1)` scriverebbe `111 222 333 ...`

Beep

Questa istruzione emette il suono di base stabilito dal sistema operativo.

Beep Frequenza Durata

Questa istruzione emette un suono sinusoidale puro.
- La frequenza è regolabile da 37 Hz a 32700 Hz
- La durata è regolabile da 1 millisecondo a 2 miliardi di millisecondi.

Esempio che riproduce un 440 Hz per 1/10 di secondo: Beep 440 100

Beep "Stringa di caratteri"

Esempio: Beep "220 500, 440 50, 0 900, 440 50, 0 900, 440 50"
I separatori validi sono: spazi, virgole, segno meno, punto e virgola e nuova riga.
Notare che per specificare le pause si imposta a zero il valore della frequenza.

Puoi anche leggere una stringa da un file come in questo esempio:
Load s1 "HappyBirthday.txt" and in the next line Beep s1

L'istruzione Beep viene eseguita in modo asincrono, per interromperla prima della fine della sua durata si utilizza un'altra istruzione Beep o una Beep 0 0

Il segnale Beep è puramente sinusoidale, ma produce dei "tick" se la durata non è un multiplo dei cicli. Si possono ridurre questi "tick" modificando sperimentalmente le durate, eventualmente utilizzando l'apposita applicazione di esempio.

- - - - -

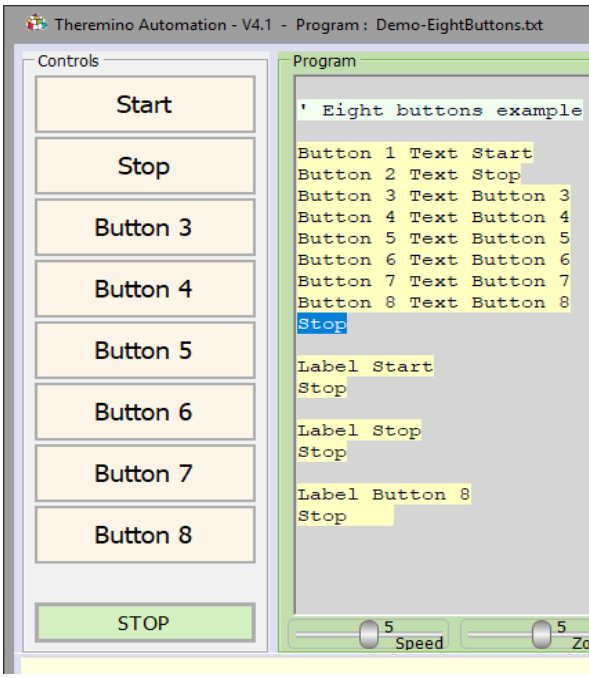
Per emettere suoni particolari, ad esempio una sirena o il click dei tasti, si può utilizzare l'istruzione LOAD (vedere come la si utilizza nelle prossime pagine).

Per emettere suoni complessi e regolabili, si possono utilizzare altre applicazioni thereminiche, ad esempio ***Audio Generator*** oppure ***Sound Player*** o anche ***Theremin Synth***, e controllarle da Automation, con l'istruzione SLOT.

Per controllare le applicazioni esterne con Automation si scrivono i valori di controllo negli Slot. Ad esempio per Audio Generator si potrebbe utilizzare lo Slot uno per la forma d'onda, il due per la frequenza, il tre per l'ampiezza e il quattro per il Pan (posizione nello stereo).

Vedere anche gli esempi nella cartella “Demo Programs / Demo Audio”

Button



I pulsanti sono sul lato sinistro della finestra e si premono con il pulsante sinistro del mouse, oppure con un dito se si utilizza uno schermo tattile.

Si possono far apparire fino a 120 pulsanti, su più colonne, e stabilire, per ognuno di essi, il testo da visualizzare.

Se esiste una istruzione LABEL con il testo corrispondente, allora premendo il pulsante, l'esecuzione del programma continuerà dalla linea della LABEL.

Se non esiste una LABEL corrispondente allora la riga viene marcata con un colore arancione e nella barra inferiore appare un messaggio di errore.

A volte può essere utile avere pulsanti non associati a una funzione, ad esempio per utilizzarli come etichette.

In questi casi si può racchiudere il testo con doppi apici per eliminare il controllo degli errori.

Il salto alla LABEL è di tipo GOSUB. Per cui, se si incontrerà un RETURN, l'esecuzione riprenderà da dove era stata interrotta. Altrimenti continuerà senza più tornare.

Se il testo è composto da più parole allora per la LABEL corrispondente si può usare anche solo la prima parola.

Button - Identificare le etichette corrispondenti

Premendo un Button quando il programma è fermo, il testo del programma scorre fino a evidenziare la LABEL corrispondente.

Quando il programma è avviato si può premere CTRL e poi il pulsante (anche quelli disabilitati). Per utilizzare questa possibilità si deve disattivare lo scroll (pulsante Scroll-enabled).

Button Text

Normalmente la parola che si scrive dopo a `Text` serve per associare il pulsante a una funzione, ma la si può utilizzare anche come etichetta.

Se il testo è molto lungo, verrà usato un carattere più piccolo e, se necessario, il testo verrà scritto su più righe. Si consiglia comunque di non usare parole o frasi troppo lunghe per evitare di non visualizzare correttamente il testo sui bottoni.

Si può forzare il testo su due righe aggiungendo un CRLF in mezzo al testo, come in questo esempio: `Button B1 Text "Prima riga" + CRLF + "Seconda riga"`

Per rendere invisibile un pulsante si scrive un testo vuoto (solo due doppi apici). Per renderlo visibile ma senza testo, si scrive uno spazio tra doppi apici.

Per visualizzare tutti i pulsanti la finestra deve essere abbastanza alta, altrimenti gli ultimi pulsanti in basso non vengono visualizzati.

Button Disabled / Enabled

Per disabilitare un pulsante si scrive: `Button <identifier> Disabled`

Per abilitare un pulsante si scrive: `Button <identifier> Enabled`

Button Slot

Con la parola Slot si collega il pulsante a uno Slot.

Per cui scrivendo, ad esempio: `Button <identifier> Slot 6`, se il valore dello Slot supera il 500, allora il pulsante viene premuto.

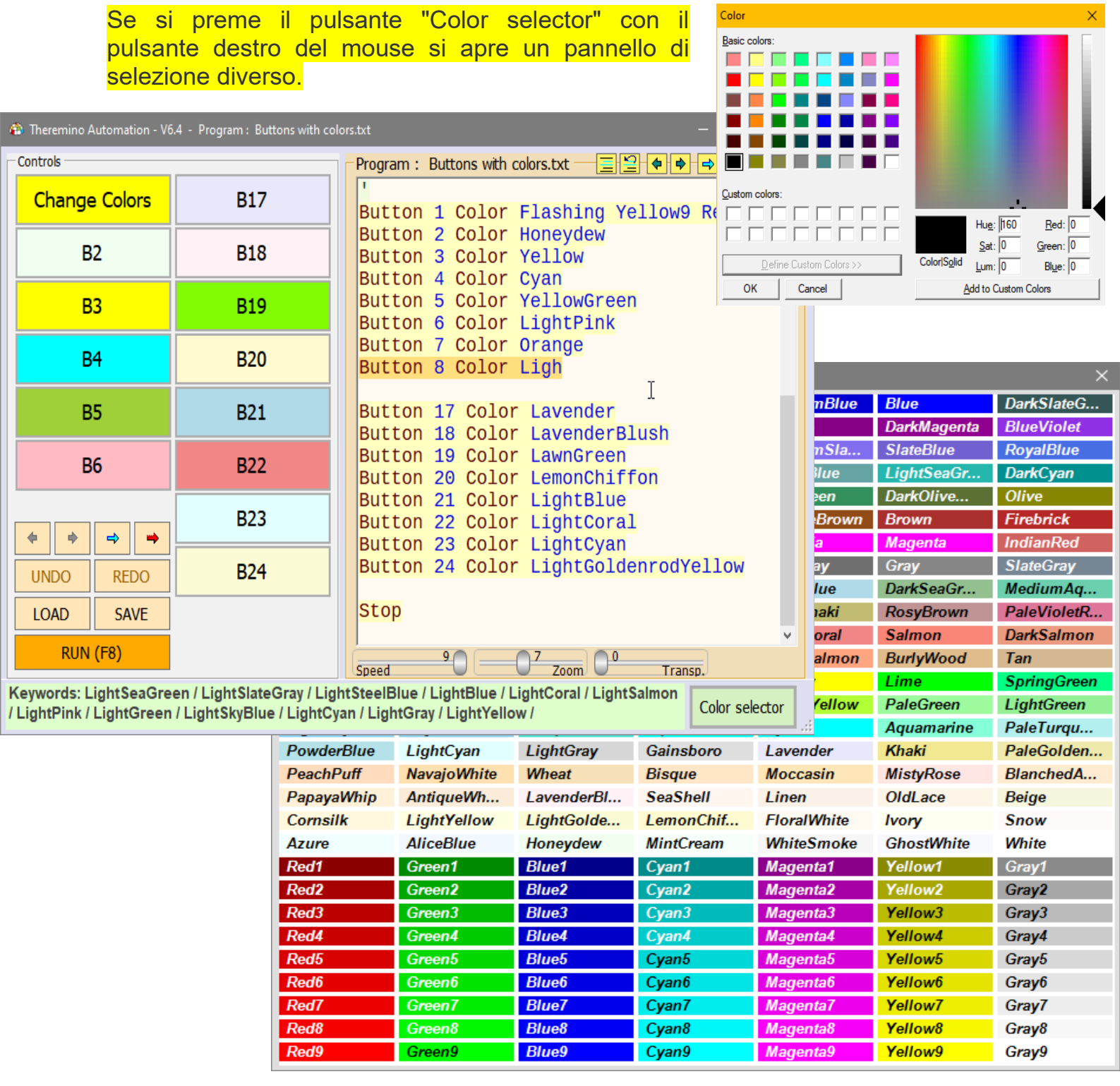
Vedere anche gli esempi nella cartella
"Demo Programs / BUTTONS"

Button Color

L'struzione "Button Color" colora i pulsanti per renderli più significativi e visibili.

I colori si scelgono scrivendo il loro nome, oppure scrivendo la lettera iniziale e poi facendo click sui nomi che vengono proposti nella barra inferiore. Ma si può anche premere il pulsante "Color selector" e sceglierli cliccando con il mouse.

Se si preme il pulsante "Color selector" con il pulsante destro del mouse si apre un pannello di selezione diverso.



Button - Colori Flashing e RGB

Colori Flashing

Con la istruzione "Button xxx Color Flashing" si possono impostare due colori e farli lampeggiare alternativamente. Con Flashing1, Flashing2... fino a Flashing9, si può anche regolare la velocità del lampeggio da molto lento a molto veloce.

Ecco un esempio di come impostare due colori che si alternano velocemente.

```
Button B1 Color Flashing7 Yellow9 Red7
```

Colori RGB

Al posto dei nomi dei colori è anche possibile scrivere la quantità di Rosso, Verde e Blu che li compongono.

La sintassi può essere ad esempio: Button xxx Color R,G,B dove al posto di R, G e B si scrivono tre numeri da 0 a 255.

Ecco alcuni esempi:

```
Button B1 Color 255,0,0     ' Questo è un rosso puro
Button B1 Color 255,255,0   ' Questo è un giallo
Button B1 Color 200,200,200 ' Questo è un grigio chiaro
```

Colori ARGB

Se si aggiunge un quarto numero al colore RGB allora il colore diventa di tipo ARGB, con al primo posto il valore di Alpha (trasparenza).

Il primo dei quattro valori è la trasparenza e va da 0 (completamente trasparente) a 255 (completamente opaco) con tutte le vie di mezzo.

Ecco alcuni esempi:

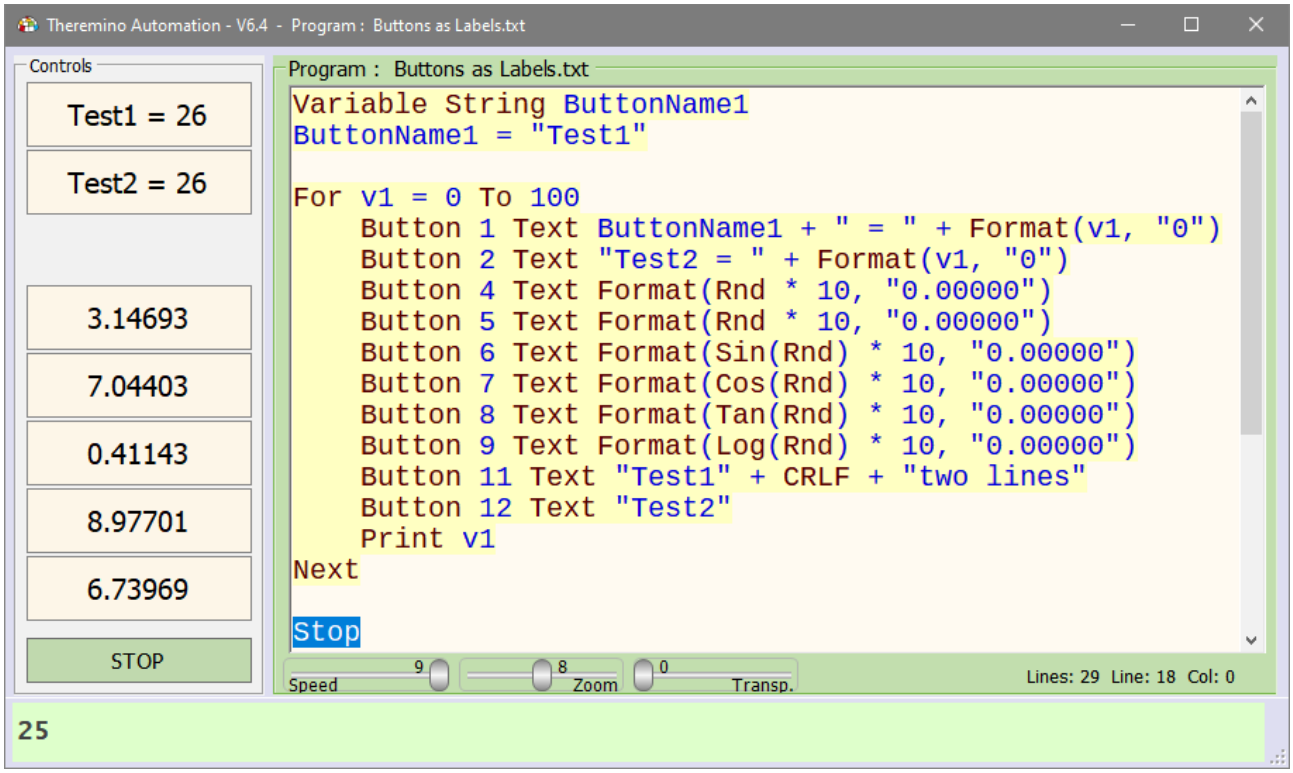
```
Button B1 Color 20,255,0,0   ' Questo è un rosso molto sbiadito
Button B1 Color 10,255,255,0 ' Questo è un giallo chiarissimo
```

\- - - -

Ci sono esempi per i Button con Immagini e con colori Flashing, RGB e ARGB nei file "Button IMAGES.txt", "Buttons Flashing.txt" e "Button RGB COLORS.txt" che si trovano nella cartella "\Demo Programs\BUTTONS"

Button usati come etichette

Si possono utilizzare i "Button" anche per visualizzare valori numerici o messaggi di testo.



Il testo dei pulsanti viene aggiornato velocemente, per cui lo si può modificare in vari punti del programma per mostrare l'evolversi dei valori numerici o per modificare il testo quando si preme il pulsante.

Importante notare che i pulsanti rimangono utilizzabili anche per chiamare funzioni del programma. A questo scopo, basta che la prima parola (prima del primo spazio) sia uguale alla parola della Label.

Se un bottone non è utilizzato per chiamare una Label, si devono utilizzare le doppie virgolette nel testo della sua dichiarazione per nascondere il relativo errore.

Ad esempio, premendo i primi due pulsanti della immagine di questa pagina verranno chiamate le funzioni che iniziano con Label Test1 e Label Test2, anche se il testo seguente (= 26) viene modificato.

Si possono anche presentare messaggi colorati e lampeggianti, per renderli più evidenti.

Vedere anche gli esempi nella cartella
"Demo Programs / BUTTONS"

Button Text e Identificatori

Quando si usano i Button come etichette con un testo che può variare, si deve dichiarare il pulsante con un identificatore (cioè con un testo senza spazi e senza usare le virgolette) e successivamente cambiare il testo con una stringa.

Questo è molto utile anche per identificare due diversi pulsanti che contengono del testo che inizia con la stessa parola, che altrimenti il programma confonderebbe nella loro esecuzione.

Ad esempio:

```
Button 1 Label Function1
Button Function1 Text “Run function 1”

Button 2 Label Function2
Button Function2 Text “ Run function 2”
Button Function2 Color White
Button Function2 Disabled

Label Function1
    ...
Return

Label Function2
    …
Return
```

Nell'esempio, la prima riga assegna al pulsante 1 l'identificatore Function1.

L'identificatore viene poi usato al posto del numero del Button per cambiare il testo mostrato sul pulsante con una stringa (seconda riga dell'esempio), oppure per chiamare la Label corrispondente. Così l'identificatore rimane sempre lo stesso anche se il testo del pulsante cambia.

Anche un Button dichiarato attraverso una stringa di testo (racchiusa tra doppi apici) può svolgere lo stesso ruolo, ma questo può essere fonte di ambiguità in alcuni casi.

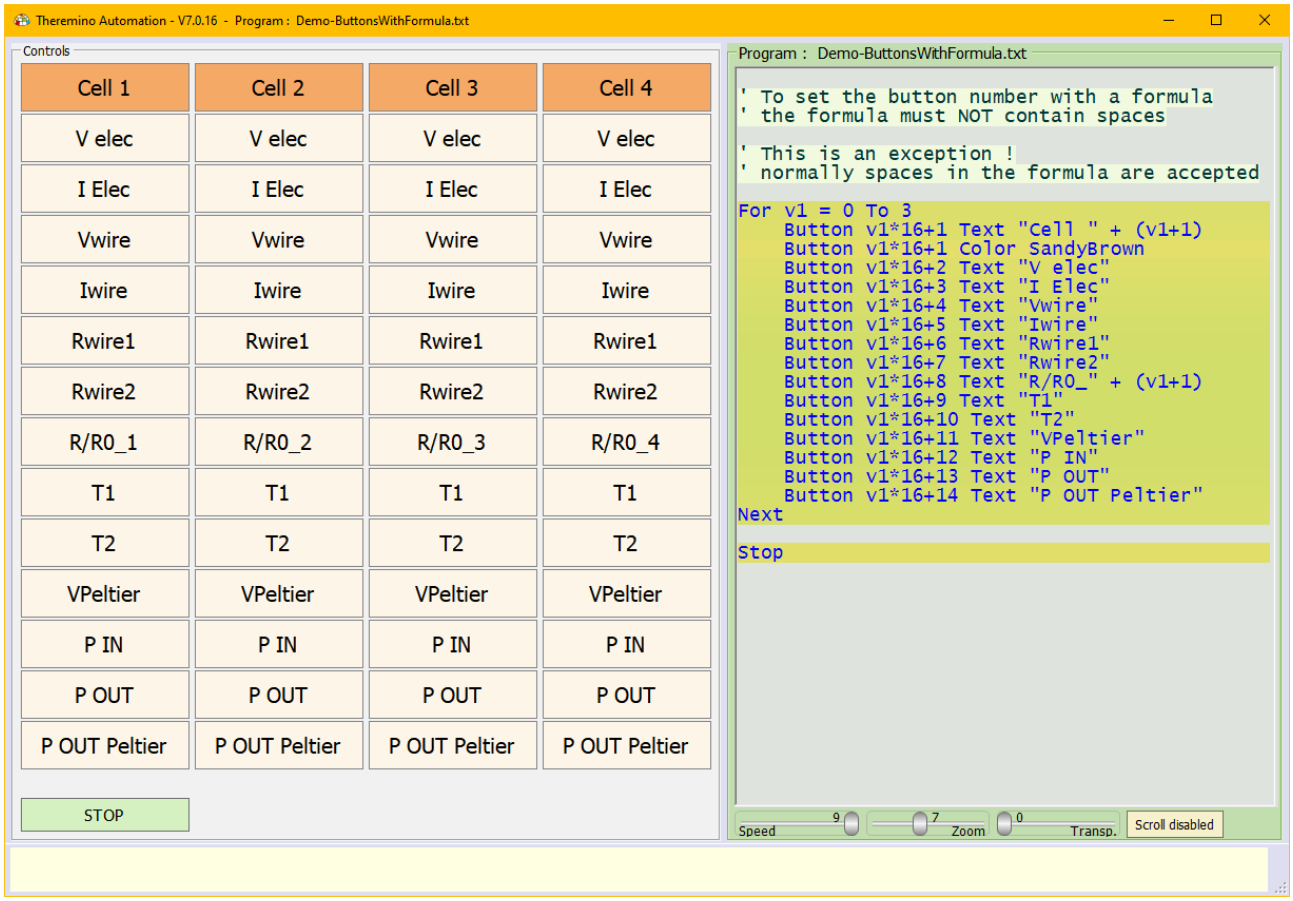
La cosa più importante da tenere a mente è che un Button verrà identificato in tutto il programma attraverso l'identificatore o la prima parola di una stringa che si è utilizzati nella sua **prima** dichiarazione all'interno del programma.

Se si scrive Button nn Text xxx la parola Text verrà sostituita automaticamente con Label non appena si ricarica il programma.

Button - Formule al posto dei numeri

Normalmente si scrive un numero per ogni pulsante e in questo modo li si possono spostare di posizione facilmente senza cambiare nient'altro nel programma.

In alcuni casi potrebbe essere utile scrivere una formula al posto del numero e racchiudere il tutto in un ciclo FOR come si vede nella immagine seguente.



Da notare che in questo caso le formule non devono contenere spazi.
Ma questa eccezione vale solo per indicare i numeri dei Button.
Normalmente le formule possono anche contenere spazi tra i termini.

- - - - - -

Vedere anche l'esempio:
"Demo Programs \ BUTTONS \ ButtonsWithFormula.txt"

Button che cambiano di stato

L'esempio seguente crea un pulsante che accende e spegne un LED.

Ogni volta che si preme il pulsante il testo del pulsante e il suo colore cambiano e viene anche modificato il valore dello Slot 1 che controlla la luminosità del LED.

```
' ---------------------------------- INITIALIZATIONS
Variable Numeric LED = 1
Button 1 Text LedToggle
Gosub LedOFF
Stop

' ------------------------------------ FUNCTIONS
Label LedToggle
     If Slot(LED) = 0
          Gosub LedON
     Else
          Gosub LedOFF
     EndIf
Return

Label LedON
     Slot LED = 1000
     Button LedToggle Text "LED OFF"
     Button LedToggle Color PowderBlue
Return

Label LedOFF
     Slot LED = 0
     Button LedToggle Text "LED ON"
     Button LedToggle Color Cyan
Return
```

Con questa struttura si possono anche chiamare la tre funzioni da varie parti del programma (con `Gosub LedToggle`, `Gosub LedON` e `Gosub LedOFF`) e in tutti i casi il testo e i colori del Button vengono aggiornati correttamente.

Volendo è anche possibile espandere questo meccanismo ad un numero maggiore di stati. In questo caso la funzione "Toggle" si chiamerà "Change" e dentro ad essa si utilizzeranno delle istruzioni "Case" per passare da uno stato al successivo.

Sperimentate con gli esempi "ButtonBistable" e "ButtonMultipleStates"
nella cartella “Demo Programs \ Buttons”

Button con immagini

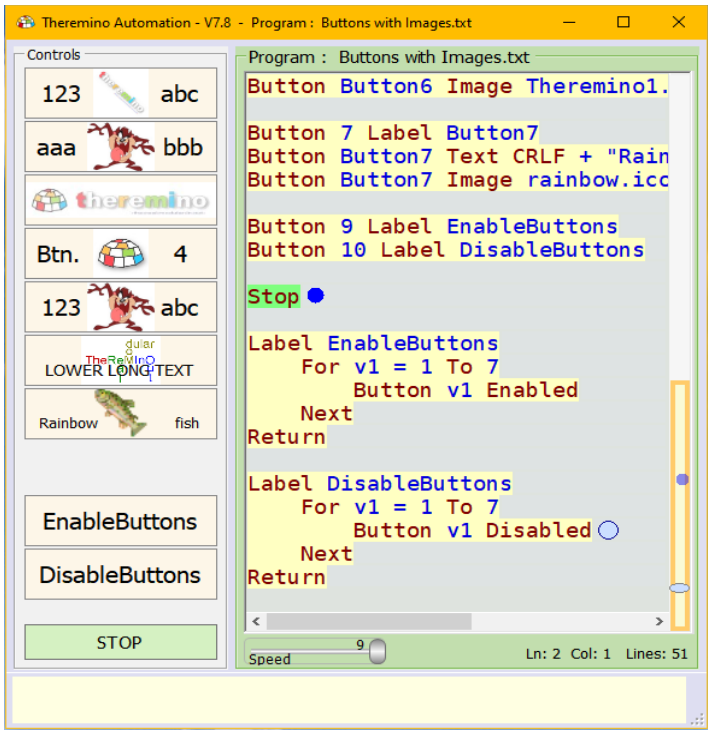
Per assegnare immagini ai pulsanti si scrive, ad esempio:

```
Button 1 Label Button1
Button Button1 Text "123   abc"
Button Button1 Image Image1.jpg
```

In questo caso la immagine è una JPG ma potrebbe anche essere una PNG, ICO, GIF, BMP, o anche altri formati che il sistema operativo riconosce.

Le immagini devono trovarsi nella cartella "Media" che trovate vicino al file "Automation.exe".

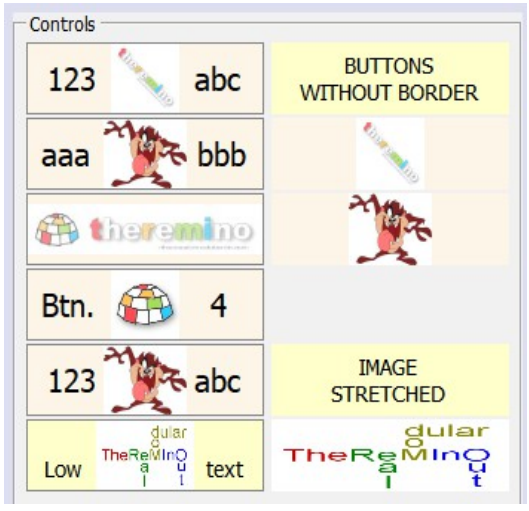
Se si trovano in una sotto-cartella bisogna aggiungere la parte di percorso mancante.



Si possono anche utilizzare immagini rettangolari con la parte sinistra bianca e posizionare il testo a sinistra. Oppure immagini con la parte destra bianca e utilizzare spazi ripetuti per spingere il testo a destra.

Quando si disabilitano i pulsanti le immagini vengono automaticamente trasformate in toni di grigio.

Questo aiuta a individuare i pulsanti abilitati e distinguerli meglio da quelli disabilitati.



Quando si impostano le immagini si può anche eliminare il bordo nero scrivendo "Image0":

```
Button Button1 Image0 Image1.jpg
```

Oppure si può renderlo più spesso scrivendo `Image2`, `Image3`, `Image4` o `Image5`.

E infine si può anche stirare la immagine fino ai bordi scrivendo `Image0S` (zero significa senza bordo e S significa "Stretched")

Vedere l'esempio “Demo Programs \ BUTTONS \ Button IMAGES.txt”

Comandi COM (porta seriale)

COM Open 1 9600 oppure **COM Open COM1 9600**
Questi comandi aprono la porta seriale "COM1" alla velocità di 9600 baud. Al posto di 9600 si può utilizzare qualunque velocità da 1 baud fino a molti mega baud, limitati solo dall'hardware usato.

Per conoscere i nomi delle porte valide, si utilizza la funzione **COM_Portnames** spiegata nella prossima pagina.

Per sapere se la porta è effettivamente aperta si utilizza la funzione **COM_Status** spiegata nella prossima pagina.

COM Close
Questo comando chiude la porta seriale. Si può usarlo anche se la porta non è aperta, per assicurarsi che sia chiusa.

COM WriteString s1
COM WriteLine s1
Questi due comandi inviano la stringa s1 alla porta seriale.
Il primo invia solo la stringa, mentre il secondo aggiunge un CRLF finale.

COM WriteStringHEX s1 per esempio "AA AA FF 80 00"
COM WriteStringDEC s1 per esempio "170 170 255 128 0"
Questi due comandi inviano sequenze di molti byte specificati da numeri esadecimali o decimali. I numeri devono essere separati da spazi (non virgole o altri separatori) e gli unici caratteri validi sono da "0" a "9" e da "A" ad "F".

COM DiscardOutBuffer
COM DiscardInBuffer
Questi comandi svuotano i buffer di uscita e di ingresso della porta seriale.
Svuotare i buffer può essere utile in alcuni casi, per azzerare la storia passata quando si inizia una nuova sessione di comunicazione.

COM EnableDTR
COM EnableRTS
COM DisableDTR
COM DisableRTS
Questi comandi alzano e abbassano le uscite DTR e RTS.
Alcuni dispositivi rispondono solo se queste due uscite sono in un certo stato.

- - - - - -

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo COM Port”

Funzioni COM (porta seriale)

`S1 = COM_Status`

In questo esempio la variabile S1 vale "OPENED" se il comando "Open" ha potuto aprire la porta, oppure "CLOSED", se non la porta non è stata aperta.

I motivi per cui non si riesce ad aprire la porta, potrebbero essere che il numero di porta richiesto non esiste sul dispositivo, o che è attualmente utilizzato da un altra applicazione.

`S1 = COM_Portnames`

In questo esempio la variabile S1 viene riempita con i nomi delle porte utilizzabili separati da spazi.

Ecco un esempio di stringa fornita da questa funzione **`"COM1 COM12 COM13"`**

Interessante notare che posizionando il cursore sulla riga **`S1 = COM_Portnames`** le porte vengono visualizzate, anche senza eseguire il programma. Questo è un comodo comportamento utilizzabile per tutte le funzioni che impostano una variabile.

Per contare il numero di porte dentro a questa stringa e per scegliere uno dei nomi, si possono usare le funzioni **GetSeparatedStringCount** e **GetSeparatedString** che sono spiegate in ***questa pagina***

`COM_Dsr`

`COM_Cts`

Queste funzioni leggono lo stato dei segnali DSR e CTS. Il risultato è un valore Booleano per cui lo si testa direttamente nelle istruzioni IF come le seguenti: **`If COM_Dsr`** oppure **`If COM_Dsr = True`** oppure **`If COM_Dsr = False`**

E' anche bene ricordare che assegnando valori booleani a variabili numeriche, poi si dovrà testarle per ZERO (0 = False) o per NOT ZERO (-1 = True).

- - - - - -

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo COM Port”

Buffer di ricezione della COM (porta seriale)

`S1 = COM_Received`

Questo esempio legge una riga di caratteri in arrivo dalla porta seriale.

Non si ricevono i singoli caratteri ma solo righe complete. La ricezione avviene solo quando il dispositivo che trasmette invia un carattere di fine linea.

La fine della linea può essere specificata da una qualunque delle seguenti combinazioni di caratteri: **CR** (13), **LF** (10), **CRLF** (13-10), oppure **LFCR** (10-13).

La stringa ricevuta non contiene i caratteri di fine linea.

`S1 = COM_ReceiveUntil(string)`

Questo esempio legge un certo numero di caratteri in arrivo dalla porta seriale.

La fine della linea può essere specificata dai caratteri di terminazione, che si forniscono nel parametro "string".

Come terminazione si può utilizzare un singolo carattere o più caratteri.

La stringa ricevuta contiene anche i caratteri di terminazione.

Per specificare caratteri di terminazione non stampabili (inferiori al 32 o superiori al 127) si può utilizzare la funzione Chr().

Ad esempio per specificare come terminatore i caratteri zero e 255 si potrebbe scrivere: `Chr(0) + Chr(255)`

`S1 = COM_ReceiveNChars(nn)`

Questo esempio legge un certo numero di caratteri in arrivo dalla porta seriale.

Il numero di caratteri da ricevere è specificato dal valore numerico "nn"

Se il buffer della seriale non contiene caratteri o se contiene meno caratteri di quelli richiesti allora questa funzione risponde una stringa vuota.

- - - - -

La stringa di caratteri ricevuta può essere scandita con la funzione Mid e convertita in vari modi, ad esempio le righe seguenti convertono i caratteri in valori esadecimali e stampano: 01 03 50 80 BF FF

```
s1 = Chr(1) + Chr(3) + Chr(80)+ Chr(128) + Chr(191) + Chr(255)
For v1 = 1 To Len(s1)
     s2 = s2 + Right("0" + Hex(Asc(Mid(s1, v1))), 2) + " "
Next
Print s2
```

Controls

Probabilmente nelle future versioni aggiungeremo altri controlli, ma attualmente tra i controlli c'è solo un comando per impostare la posizione del cursore del mouse e una TextBox (casella di testo), utilizzabile per visualizzare molte linee di testo con il comando Print.

Le istruzioni "Controls" sono:

```
Controls SetCursorPos <X Y>
```

```
Controls SetBackColor <Color>
```

```
Controls OpenTextBox
```

```
Controls CloseTextBox
```

```
Controls ClearTextBox
```

I dettagli di queste istruzione sono spiegati nelle prossime sezioni.

Controls - SetCursorPos <X Y>

Questa istruzione posiziona il cursore del mouse sullo schermo.

I valori numerici X e Y sono in pixel dello schermo, per cui a sinistra in basso i due valori valgono zero, mentre a destra in alto i due valori dipendono dal numero di pixel dello schermo.

In caso di schermi multipli i pixel iniziano in basso a sinistra del primo schermo fino a in alto a destra dell'ultimo schermo e se lo schermo principale non è il primo allora i numeri che identificano i pixel possono anche essere negativi.

Si può anche leggere la posizione del cursore del mouse
con le funzioni MouseXP and MouseYP spiegate ***in questa pagina***.

Vedere anche gli esempi nella cartella
"Demo Programs \ Demo Mouse"

Controls - SetBackColor <color>

Questa istruzione colora lo sfondo del riquadro dei Button con un colore a scelta o con un colore specificato con valori RGB.

Si può anche scegliere il colore con il pulsante "Color selector" e se lo si preme con il pulsante destro del mouse si apre un pannello di selezione diverso.

Controls - TextBox

Per controllare la TextBox si utilizzano i seguenti comandi:

`Controls OpenTextBox`

Con questo comando si apre la TextBox e tutti i successivi "Print" verranno visualizzati nella TextBox al posto che nella riga di stato inferiore.

Quando la TextBox è aperta si può ridimensionarla muovendo con il pulsante sinistro del Mouse la barra grigia orizzontale che la divide dall'area del programma (vedere l'immagine della prossima pagina).

`Controls CloseTextBox`

Con questo comando si chiude la TextBox e tutti i successivi "Print" verranno visualizzati nuovamente nella riga di stato inferiore.

`Controls ClearTextBox`

Con questo comando si elimina tutto il testo presente nella TextBox.

Si può cancellare il testo anche inviando un CLS o un Chr(12) con l'istruzione Print, come nei due esempi seguenti:

```
Print CLS
```

```
Print Chr(12)
```

La lunghezza della TextBox è limitata a diecimila caratteri per evitare rallentamenti. Quando si supera questa dimensione i caratteri vengono eliminati automaticamente dalla parte superiore della TextBox e sono sostituiti con una riga di avvertimento.

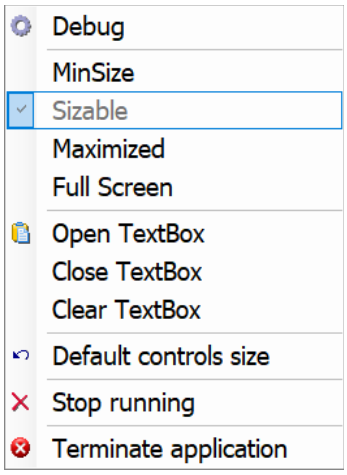
- - - - -

Vedere anche gli esempi nella cartella
"Demo Programs \ Demo TextBox"

Controls - Chiudere e ridimensionare la TextBox

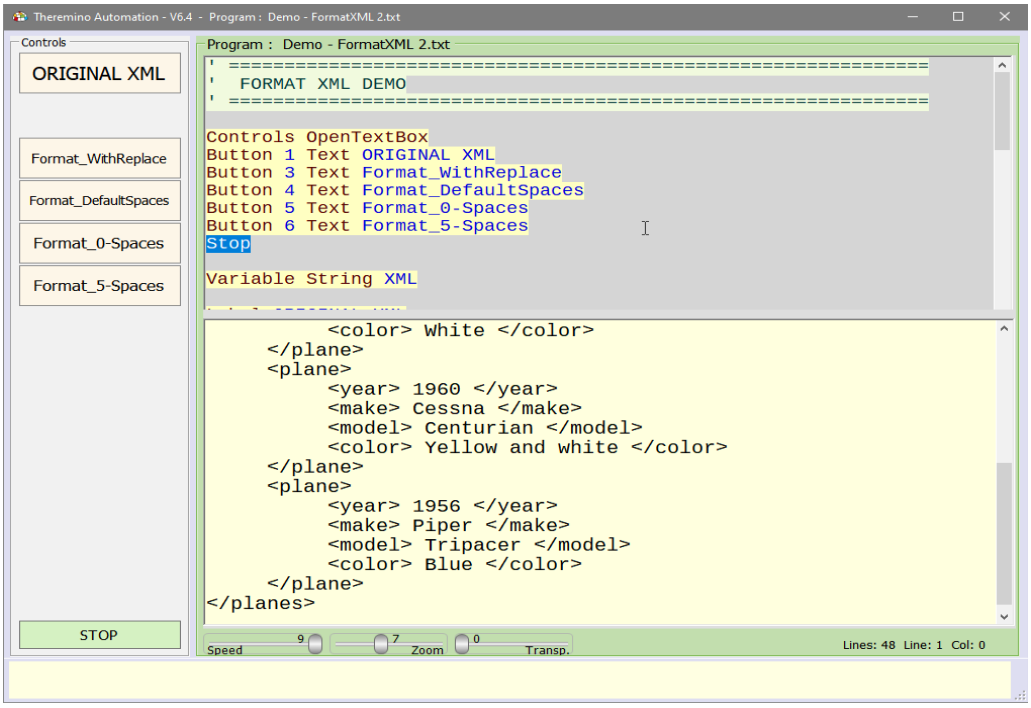
La TextBox si apre eseguendo una `Controls OpenTextBox`. Per chiuderla automaticamente quando si ferma il programma, si può scrivere l'istruzione `Controls CloseTextBox` nell'evento di chiusura, come nell'esempio seguente:

```
Label EventStop
     Controls CloseTextBox
End
```



Normalmente si controlla la TextBox per mezzo di istruzioni nel programma, ma in alcuni casi potrebbe essere utile controllarla manualmente.

Premendo il tasto destro del mouse sulla parte sinistra della applicazione si apre un menu contestuale che permette di Aprire, Chiudere e Svuotare la TextBox anche quando il programma è fermo.



Per ridimensionare la TextBox (che si vede nella metà inferiore di questa immagine), si sposta con il Mouse la barra grigia orizzontale che la divide dall'area del programma.

For - Next

Queste due istruzioni eseguono ripetutamente le linee comprese tra FOR e NEXT.

L'istruzione FOR, che deve stare prima del NEXT, è composta da una variabile numerica (**counter**), un valore iniziale (**initial**), e un valore finale (**final**).

```
For counter = initial To final
   ---
   ---
Next
```

Quando si esegue il FOR arrivando dalle istruzioni precedenti, la variabile "**counter**" viene impostata con il valore "**initial**".

Dopo aver eseguito le istruzioni che stanno nelle linee seguenti al FOR, il programma arriva al NEXT. A questo punto, se la variabile "**counter**" è ancora diversa dal valore "**final**", allora l'esecuzione del programma torna al FOR.

Quando si ripete il FOR la variabile "**counter**" viene incrementata o decrementata di uno, a seconda che il valore "**final**" sia maggiore o minore di essa. La direzione dell'incremento è automatica. Se si vuole specificare la direzione allora si aggiunge la parola "**Step**" spiegata nella prossima pagina.

Se per qualche motivo la variabile "**counter**" superasse il valore "**final**" (ad esempio perché si sono utilizzati valori numerici non interi, o per errori di arrotondamento), allora verrebbe impostata al valore "**final**". In questo modo si garantisce che l'ultimo giro venga sempre eseguito con il valore "**final**".

I valori "**initial**" e "**final**" possono essere delle variabili, oppure delle costanti numeriche, o anche delle espressioni, come negli esempi seguenti:

```
For var1 = var2 To var3
For var1 = 7 To 3
For var1 = var2+7 To "var3 - 4 + int(rnd * 6)"
```

Importante notare che le espressioni non devono contenere spazi, altrimenti la riga diventa rossa e la istruzione FOR non viene eseguita.

Si possono scrivere espressioni complesse, anche contenenti spazi e funzioni, racchiudendole tra doppi apici, come visibile nella parte destra dell'ultimo esempio.

For - Next con Step

Normalmente ad ogni ciclo FOR e NEXT la variabile di controllo viene incrementata o decrementata di una unità.

Aggiungendo la parola **Step** e una espressione numerica è possibile imporre un incremento di una frazione di unità, o anche un incremento più grande, come nei prossimi esempi.

```
For v1 = 1 To 9 Step 0.1
   ---
Next
```

```
For v1 = 0 To 32000 Step 1000
   ---
Next
```

In questi esempi come variabile di controllo abbiamo usato **v1** e come valori iniziali e finali dei numeri, ma naturalmente si possono usare anche variabili definite dall'utente e espressioni complesse.

Usando **Step** la direzione dell'incremento non è più automatica ma la si specifica esplicitamente, come nei due esempi seguenti.

```
For v1 = 9 To 1 Step -0.1
   ---
Next
```

```
For v1 = 32000 To 0 Step -1000
   ---
Next
```

Se si specificano i valori di inizio e fine con variabili, il **For** potrebbe incrementare nella direzione imprevista. Nell'esempio seguente, che produce tre brevi suoni, ci si aspetterebbe che cambiando **"abc"** in **""** il **For** non venisse eseguito, ma inaspettatamente esegue i valori uno e poi zero e quindi emette due suoni.

```
s1 = "abc"
For v1 = 1 To Len(s1)
   Beep 1000 500
Next
```

In questi casi è bene specificare lo **Step** e stabilire la direzione dell'incremento.

For - Next al posto di While

In altri linguaggi si utilizzano molti tipi di cicli, ad esempio While-Wend, Do-While, Do-Loop, Loop-Until, Loop-While, ecc..

Questi molteplici modi di scrivere i loop fanno tutti esattamente la stessa cosa, per cui noi ne useremo uno solo, il For-Next.

Il For-Next è senz'altro il metodo più versatile e completo per costruire un ciclo di ripetizione e può anche produrre cicli infiniti se si imposta come valore finale il numero 9e99 (o "Infinity"), come negli esempi seguenti:

```
For v1 = 1 To 9e99
   ---
   ---
Next
```

Per creare cicli infiniti si può utilizzare la funzione "Infinity" al posto di 9e99.
Per terminare il ciclo, si può impostare la variabile di controllo con il valore "Infinity", oppure si potrebbe utilizzare la parola "Exit", come spiegato nelle prossime pagine.

```
For v1 = 1 To Infinity
   If Slot(1) > 500
      v1 = Infinity
   EndIf
Next
```

I For-Next possono anche essere annidati e si può uscire facilmente anche dai cicli For-Next più interni con l'istruzione Exit, come nell'esempio seguente:

```
For v8 = 1 To 10
   For v7 = 1 To 30
      For v6 = 1 To 50
         ---
         If Slot(1) > 500
            Exit 3
         EndIf
      Next
   Next
Next
```

Le prossime due pagine spiegano come usare l'istruzione Exit

Exit

Questa istruzione deve essere inserita in un ciclo For-Next e fa saltare l'esecuzione del programma alla istruzione successiva al **Next**.

Ecco un esempio di uscita prematura da un For-Next

```
For v1 = 1 To 1000
     If v1 >= 10
          Exit
     EndIf
Next
```

Senza l'istruzione **Exit** questo ciclo si ripeterebbe mille volte, ma la istruzione **If** fa terminare il ciclo quando **v1** arriva a 10.

In altri casi al posto di controllare **v1** si potrebbe uscire quando viene premuto un tasto, ad esempio "ESC", oppure se è passato troppo tempo.

Questo esempio mostra che i cicli For-Next possono anche essere annidati.

```
For v1 = 0 To 1000
     If v1 >= 10
          Exit
     EndIf
     For v2 = 0 To 1000
          If v2 >= 20
               Exit
          EndIf
     Next
Next
```

In caso di cicli For-Next annidati ogni istruzione Exit agisce sul ciclo più interno.

Per cui in questo esempio:

- Il primo **Exit** si riferisce al **For v1** e quindi esce dal **Next** finale.
- Il secondo **Exit** si riferisce al **For v2** e quindi esce solo dal suo **Next**.

L'istruzione Exit ha effetto solo se inserita in un ciclo For-Next. Se la si posiziona al di fuori di esso viene evidenziata in colore arancio e non viene eseguita.

Vedere anche gli esempi nella cartella: "Demo Programs\Demo FOR NEXT"

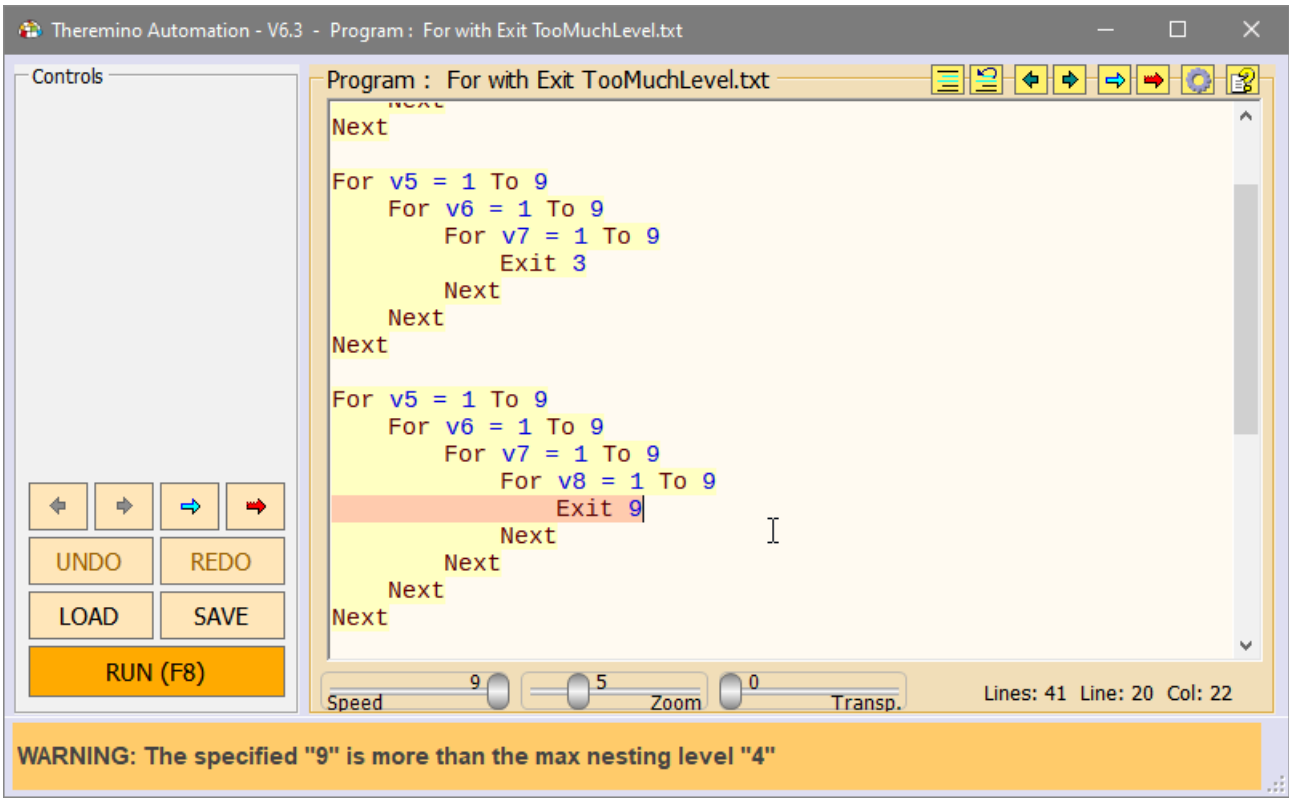
Exit "n"

Questa istruzione permette di uscire da più For-Next annidati uno dentro l'altro.

```
For v5 = 0 To 9
     For v6 = 0 To 9
          Exit 2
     Next
Next
```

```
For v5 = 1 To 9
     For v6 = 1 To 9
          For v7 = 1 To 9
               For v8 = 1 To 9
                    Exit 4
               Next
          Next
     Next
Next
```

Se si specifica un numero maggiore del massimo livello di For-Next annidati allora l'istruzione Exit non viene eseguita e si viene avvertiti dell'errore con un messaggio come nella immagine seguente:



Goto - Gosub - Return

Le istruzioni GOTO e GOSUB fanno saltare l'esecuzione del programma alla linea indicata con una LABEL (etichetta) e con un testo identificativo, composto da caratteri alfanumerici.

- Quindi l'istruzione **`Goto xxx`** salterà alla linea indicata con **`Label xxx`**.
- E l'istruzione **`Gosub yyy`** salterà alla linea indicata con **`Label yyy`**.

Se non esiste la etichetta corrispondente, allora le istruzioni GOTO e GOSUB non hanno effetto e l'esecuzione prosegue come se non ci fossero.

L'identificativo può anche essere composto da più parole, e queste parole possono anche essere racchiuse tra doppi apici (per chiarezza), ad esempio si potrebbe scrivere **`Label xxx yyy zzz`**, oppure **`Label "xxx yyy zzz"`**.

Differenza tra Goto e Gosub

Per dirla in parole semplici, il GOTO è come un viaggio di sola andata, mentre il GOSUB prevede anche il ritorno.

Quindi se si utilizza un GOTO, il programma salta e continua da quella posizione. Mentre se si utilizza un GOSUB, il programma salta, esegue alcune istruzioni e, quando incontra una istruzione RETURN, torna alla istruzione dopo al GOSUB, che in questo caso è l'istruzione Stop.

```
Gosub Identifier1
Stop

Label Identifier1
...
Return
```

Vedere anche gli esempi nella cartella
"Demo Programs \ GOSUB and RETURN"

Eliminare i Gosub e inviare parametri alle funzioni

A partire dalle versioni 7.x di Automation si possono anche omettere tutti i Gosub e scrivere direttamente gli identificativi delle funzioni.
LEGGETE LE PAGINE : *Chiamare le funzioni* e *Parametri delle funzioni*

If - Else - Endif

Le istruzioni “IF” e “ENDIF” racchiudono una zona da eseguire solo se la condizione è vera.

Ogni “IF” deve terminare con il suo “ENDIF” altrimenti non viene eseguito e l'esecuzione prosegue come se non ci fosse.

```
If Slot(1) > 500
     Istruzioni da eseguire
     solo se il valore dello slot 1
     è maggiore di 500
Endif
```

Le istruzioni “IF”, “ELSE” e “ENDIF” racchiudono due zone, una da eseguire se la condizione è vera, e l'altra da eseguire se la condizione è falsa.

```
If Slot(1) > 500
     Istruzioni da eseguire
     solo se il valore dello slot 1
     è maggiore di 500
Else
     Istruzioni da eseguire
     se non è minore o uguale a 500
Endif
```

Le istruzioni IF (e anche tutte le altre strutture di Automation) possono essere “annidate”, cioè inserite una dentro all'altra. Ad esempio nell'esempio seguente il BEEP viene eseguito se tutti e tre gli Slot sono maggiori di 500.

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

Ma lo stesso risultato lo si potrebbe ottenere anche con istruzioni AND

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

- - - - - -

Vedere anche gli esempi nella cartella “Demo IF ELSE ENDIF”

Key

Con la parola chiave “Key” si ottiene l'esecuzione di parti di programma, premendo tasti della tastiera.

I tasti usabili sono i seguenti:
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, Num0, Num1, Num2, Num3, Num4, Num5, Num6, Num7, Num8, Num9, Left, Right, Up, Down, Space, Esc, PageUp, PageDown, Home, End Canc, Back, Enter, Ins, Shift, Ctrl, Alt, Print, Scroll, Pause

Le istruzioni KEY solitamente si scrivono all'inizio del programma e possono essere di due tipi GOTO o GOSUB. Con GOTO il programma salta e non torna più. Con GOSUB il programma salta e torna appena incontra un RETURN.

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

Label Identifier1
…
Stop
```

A partire dalla versione 7.x di Automation si può anche omettere la parola Gosub.

Quando si preme un tasto sulla tastiera, il programma viene interrotto, qualunque cosa stesse facendo, e salta alla LABEL corrispondente.

Se si è impostato un KEY con il RETURN, allora il programma viene interrotto, salta a eseguire le linee dopo la LABEL, poi incontra il RETURN, e torna a fare quello che stava facendo quando è stato interrotto.

Per evitare di innescarli per sbaglio mentre si scrive qualcosa, i Key non agiscono quando si naviga sulle pagine WEB, oppure quando si sta scrivendo in una casella di Input, o quando si sta scrivendo in un'altra applicazione. O in tutti i casi quando la applicazione Automation non è selezionata, o è invisibile, o è minimizzata.

Se si desidera che l'istruzione Key agisca sempre si può utilizzare l'istruzione "Option GlobalKeys Enabled".

- - - - - -

Vedere anche gli esempi nelle cartelle “Demo Keys” e "Demo Option"

Label

L'istruzione LABEL (etichetta) è un contrassegno, che si utilizza per dare un nome univoco a una linea del programma.

Il contrassegno può essere composto da un qualunque numero di caratteri alfanumerici, ma non deve contenere spazi e nemmeno i doppi apici, come negli esempi seguenti:

```
Label 1
Label 123
Label Start
Label Stop
Label MyFunction
Label My_function_with_a_long_name
```

La linea etichettata con la istruzione LABEL, può essere la destinazione delle istruzioni di salto: GOTO, GOSUB, KEY e BUTTON, come negli esempi seguenti:

```
GoTo MyFunction
GoSub MyFunction
Key 1 GoTo MyFunction
Key 1 GoSub MyFunction
Button 1 Text MyFunction
```

Vedere anche gli esempi nelle cartelle
“Demo GOSUB and RETURN” e“Demo Keys”

- - - - - -

Eliminare i Gosub e inviare parametri alle funzioni

A partire dalle versioni 7.x di Automation si possono anche omettere tutti i Gosub e scrivere direttamente gli identificativi delle funzioni.
LEGGETE LE PAGINE : *Chiamare le funzioni* e *Parametri delle funzioni*

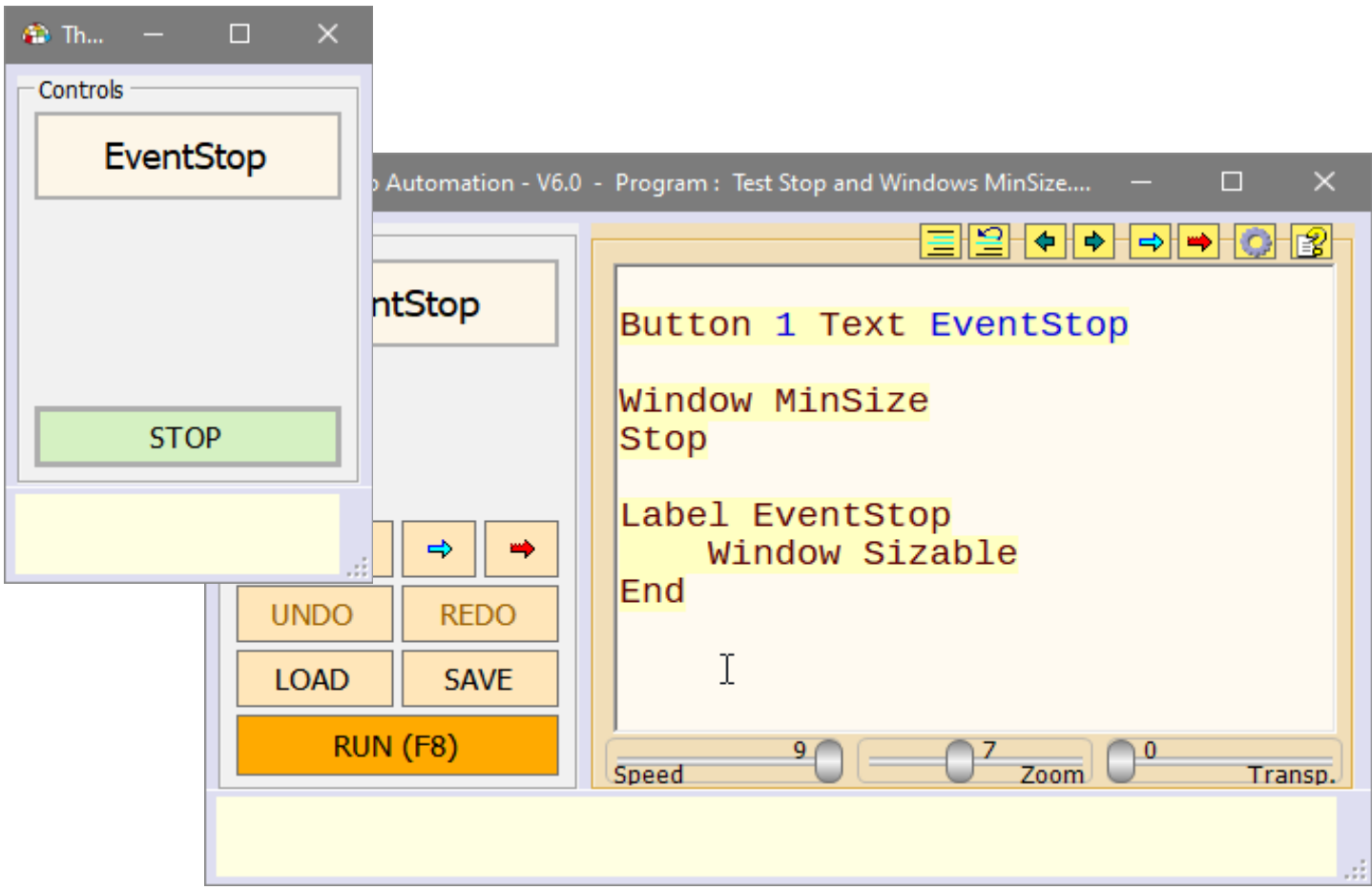
Label EventStop

L'istruzione Label EventStop viene chiamata automaticamente ogni volta che il programma viene fermato (sia manualmente, premendo il pulsante Stop, che con la istruzione End).

Questa Label è un evento speciale, ma la si può chiamare anche con Gosub EventStop, Button EventStop o Key EventStop, come si farebbe con qualunque altra Label. In questo caso il contenuto della Label viene eseguito e poi il programma si ferma perché la Label EventStop termina sempre con End.

Questo evento permette di completare azioni importanti ogni volta che il programma si ferma, ad esempio togliere corrente ai motori, ma si può usarlo utilmente in molti modi.

Nell'esempio seguente ad ogni avvio del programma la finestra viene minimizzata con il comando Window MinSize. Poi, quando si ferma il programma, viene chiamata la Label EventStop, che riporta la finestra alle dimensioni normali con il comando Window Sizable.



Vedere gli esempi nella cartella "Demo Programs \ Demo EventStop"

Label EventTimer

ATTENZIONE

La Label EventTimer viene eseguita a tempo e interrompe casualmente il programma nei punti più disparati. Possono quindi crearsi interazioni non volute, difficili da capire e da correggere.

Inoltre se la esecuzione dell'EventTimer prende troppo tempo, alcune parti del programma principale potrebbero subire notevoli rallentamenti.

Per cui usate quindi questa istruzione con molta attenzione e controllando bene che non produca effetti collaterali sul resto del programma.

- - - - -

INTERAZIONI CON IL DEBUG

Le continue interruzioni rendono difficile seguire l'andamento del programma. Per cui consigliamo di commentare l'intera funzione, da EventTimer a Return, quando si utilizzano le funzioni di Debug.

- - - - -

La Label EventTimer viene chiamata automaticamente circa dieci volte al secondo.

Le istruzioni che si trovano tra la EventTimer e il suo Return finale vengono eseguite alla massima velocità, senza riguardo per la velocità impostata con il cursore Speed. In questo modo si minimizzano gli effetti collaterali sul resto del programma.

Si consiglia comunque di osservare le seguenti regole per tutto quello che si trova tra la Label EventTimer e il suo Return finale:

- Minimizzare il numero di istruzioni.
- Possibilmente non usare istruzioni lente come Print o Button Text.
- Non usare assolutamente istruzioni molto lente come Load.
- Non utilizzare nessun tipo di attesa (Wait Seconds / Wait Button...).
- Non chiamare altre Label (soprattutto se contengono molto codice).
- Regolare Speed al massimo (9) per velocizzare l'esecuzione del programma principale. In questo modo si evita che l'esecuzione rallenti troppo o si fermi a causa delle continue interruzioni.

Label Event_DroppedFile

Si utilizza la `Label Event_DroppedFile` per ricevere il nome e il contenuto di un file che viene trascinato e rilasciato sul riquadro dei comandi con il mouse, con una operazione chiamata Drag-And-Drop.

Per il funzionamento di questo evento bisogna aver definito le variabili da leggere, `DroppedFilename` e `DroppedString`.

Se non si definisce almeno una di queste due variabili allora la riga che deve ricevere l'evento `Label Event_DroppedFile` si colora di rosso e dà un errore.

Se si vogliono leggere tutte e due le variabili le si definiscono ambedue, altrimenti basta definire solo quella che si intende usare.

Dopo aver trascinato il file la `Label Event_DroppedFile` viene chiamata e le due variabili contengono le stringhe di testo seguenti:

- `DroppedFilename` contiene il percorso completo del file più il nome e anche l'estensione.
- `DroppedString` contiene il testo completo del file. Se il testo non è leggibile, dovreste trovare una parola iniziale che spiega che il file è di un tipo speciale, ad esempio una immagine PNG o JPG.

- - - - -

Ecco un breve esempio di programma che riceve e stampa il nome del file e il suo contenuto.

```
String DroppedFilename
String DroppedString
Stop

Label Event_DroppedFile
     Print DroppedFilename + CRLF + DroppedString
Return
```

Potete provare questo esempio caricandolo dalla cartella Demo Programs \ Files.

Poi basta fare RUN dell'esempio e trascinare un qualunque file
da una cartella di Windows sulla zona "Controls".

Label Event_ExternalCommands

La Label Event_ExternalCommands viene utilizzata per ricevere comandi da qualunque applicazione in grado di scrivere negli "Slot di testo".

Quando una applicazione esterna scrive una stringa con un comando nello Slot_ExternalCommands l'evento viene notificato chiamando questa Label.

Il programma di Automation viene interrotto, qualunque cosa stesse facendo e le istruzioni dopo alla Label vengono eseguite fino al Return.

Il testo del comando si legge con la funzione "CommandText" e poi lo si decodifica con una struttura "Select", come nell'esempio seguente:

```
Variable Numeric Slot_ExternalCommands = 53
Stop

Label Event_ExternalCommands
      Select CommandText
            Case "Beep"
                  ExecBeep
                  ClearCommand
            '
            Case "Beep multiple"
                  ClearCommand
                  ExecBeepMultiple
            '
            CaseElse
                  Print "Unrecognized command: " + CommandText
                  ClearCommand
      EndSelect
Return

Label ExecBeep
      Beep 440 300
      Wait Seconds 0.5
Return

Label ExecBeepMultiple
      For v1 = 1 To 3
            Beep 880 400
            Wait Seconds 0.5
      Next
Return

Label ClearCommand
      SlotText(Slot_ExternalCommands) = ""
Return
```

Notare che nel Case "Beep multiple" è stata aggiunta una istruzione ClearCommand **prima** della esecuzione del comando. Questa istruzione dice alla applicazione esterna di proseguire immediatamente, senza attendere la fine della esecuzione del comando.

Nella prossima pagina c'è un esempio di sequenza che invia questi comandi.

I comandi da COBOT a Automation

Questo capitolo spiega come inviare comandi dalla applicazione COBOT verso la applicazione Automation.

L'esempio seguente è una sequenza che muove tre motori e tra un movimento e il successivo invia i comandi "Beep" e "Beep multiple".

```
MoveTolerance 1
MoveTime 0.5
MoveTo +00137.000 +00300.000 +00022.000
SendCommand  Beep
MoveTo +00153.000 +00303.000 +00025.000
SendCommand  Beep multiple
```

Il file che contiene questa sequenza viene eseguito dalla applicazione Theremino_COBOT e deve stare nella sua cartella "Sequences".

I comandi che si inviano alla applicazione Automation possono essere una qualunque stringa di testo, anche separata da spazi, e vengono decodificati senza tenere conto delle maiuscole o minuscole.

Eventuali caratteri TAB o Spazio multipli vengono trasformati in spazi singoli dalla applicazione COBOT prima di inviarli e i TAB e gli spazi iniziali e finali vengono eliminati.

L'utente stesso può scrivere i suoi comandi, ricordando che:
- I comandi vanno scritti nella sequenza della applicazione COBOT.
- Le istruzioni per decodificarli nell'evento ExternalCommands di Automation.

Nella applicazione Cobot si devono impostare gli Slot di testo da utilizzare per i comandi. Aprire il pannello delle opzioni (con l'ingranaggio in alto a destra) e individuare la sezione "Slots for Text-Commands" che si trova in basso.

Gli Slot di testo che si impostano nella applicazione Cobot devono essere gli stessi che si dichiarano nelle variabili di Automation con i nomi: **Slot_CommandsToCobot, Slot_CobotResponses** e **Slot_ExternalCommands.**

Vedere anche gli esempi nella cartella
"Demo Programs \ SlotText Commands"

- - - - - -

Nella prossima pagina vedremo che si possono inviare comandi anche dalla applicazione Automation verso la applicazione Cobot.

I comandi da Automation a COBOT

Per inviare comandi verso la applicazione Cobot si utilizza un metodo leggermente diverso. Si impostano le variabili **Slot_CommandsToCobot** e **Slot_CobotResponses** e poi le si utilizzano con i comandi per scrivere e leggere negli Slot di testo.

Comandi da Automation a COBOT (stringhe di testo nello **Slot_CommandsToCobot**)

- **ExitFromEdit** (esci dallo stato di "edit")
- **StartExecution** (uscita automatica dallo stato di "edit")
- **StopExecution** (uscita automatica dallo stato di "edit")
- **GotoHome** (uscita automatica dallo stato di "edit")
- **GotoCenter** (uscita automatica dallo stato di "edit")
- **EnableRepeat**
- **DisableRepeat**
- **SelectLine nnn**
- **ExecuteSelectedLine**
- **EnableCollaborative**
- **DisableCollaborative**
- **AddNewPosition**
- **SetHomePosition**
- **SetCenterPosition**
- **SetHoldingTorque nnn**
- **SetSafeTorqueLimit nnn**
- **SetTorque nnn**
- **SetAcceleration nnn**
- **SetSpeed nnn**
- **LoadSequence SeqName.txt** (il nome "xxxxx.txt" non deve contenere spazi)

Risposte dal Cobot (stringhe di testo nello **Slot_ResponsesFromCobot**)

- **Ready**
- **Execution running at line nnn**
- **Motors disconnected**
- **Human detected - Speed reduced**
- **Human too near - Execution stopped**
- **Too much torque - Execution stopped**
- **Motors disconnected - Execution stopped**

Vedere l'esempio "Commands_to_COBOT.txt" nella cartella
"Demo Programs \ SlotText Commands"

Comandi esterni verso i Button di Automation

Si possono anche eseguire i Button di Automation con comandi da applicazioni esterne.

Questo metodo è più facile da usare del metodo, spiegato nelle pagine precedenti che prevede la Label `Event_ExternalCommands`

Si invia semplicemente il testo che appare su un pulsante (o il suo identificatore) e il pulsante viene eseguito come se lo si premesse con il mouse.

- - - - -

Per utilizzare questo metodo si deve impostare lo slot dei comandi, ad esempio come in questa riga: `Variable Numeric Slot_ExternalCommands = 1`

Ma **non si deve impostare** la `Label Event_ExternalCommands`

- - - - -

FUNZIONAMENTO

Tutti i comandi che si invieranno nello `Slot_ExternalCommands` verranno confrontati, senza tenere conto di maiuscole e minuscole, con il testo presente sui pulsanti e anche con gli identificatori che si sono usati nella prima creazione dei pulsanti, e se corrispondono allora il pulsante viene premuto.

Consigliamo anche di utilizzare spazi singoli tra le parole, nel testo che appare sui Buttons da eseguire e anche di usare identificatori senza spazi, perché alcune applicazioni come ad esempio "VoiceCommands" inviano sempre spazi singoli.

- - - - -

IMPORTANTE - Per indirizzare i comandi esterni verso i Buttons
NON si deve impostare la `Label Event_ExternalCommands`

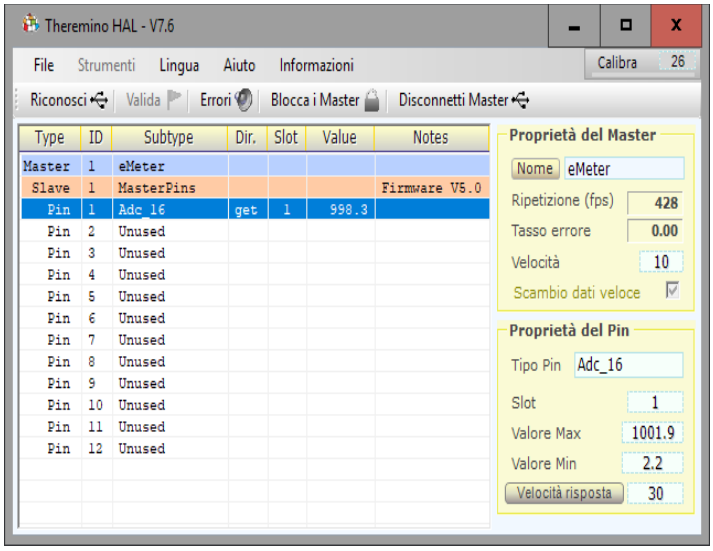
Se si imposta la `Label Event_ExternalCommands`
allora la esecuzione dei comandi esterni deve essere eseguita dentro ad essa
e i Buttons non verranno più eseguiti.

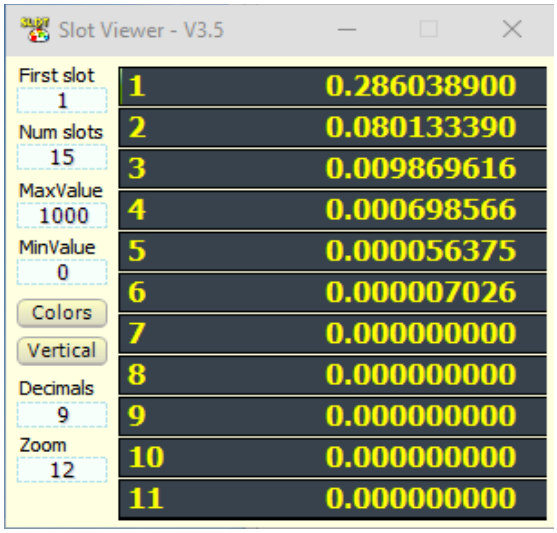
Provate l'esempio **SlotText-ExternalCommands_TO-BUTTONS.txt**
inviandogli i comandi sullo SlotText 1 con Theremino_Terminal
o con una seconda copia di Automation.

Load (Applicazioni)

In alcuni casi può essere utile avviare una applicazione tutte le volte che si esegue Automation. Spesso questa applicazione è Theremino_HAL o lo SlotViewer, o ambedue.

Se la applicazione è già attiva non viene lanciata una seconda volta.





L'istruzione Load ferma l'esecuzione del programma fino a che la applicazione è completamente avviata.

Per caricare applicazioni il loro file “xxxx.exe” dovrebbe stare nella cartella "Apps". Si potrebbe anche indicare il percorso completo, ma dovrebbe essere in un posto fisso del sistema (non nelle cartelle di Automation), altrimenti spostando Automation il Load smetterebbe di funzionare.

Se la applicazione da avviare si trova in una sotto-cartella, si può aggiungere la parte di percorso che serve, come ad esempio:
Load Apps\Theremino_HAL.exe

Se ci sono spazi nel percorso, si devono aggiungere gli apici
Load "Apps\Theremino_HAL.exe"

Se la applicazione da avviare si trova in una cartella superiore, si può salire di una cartella, con il doppio punto. Ad esempio:
Load "..\Theremino_HAL\Theremino_HAL.exe"

- - - - -

Se si specifica solo il nome del file (ad esempio **"Theremino_SlotViewer.exe"**)
allora la applicazione viene cercata nella cartella **"Apps"**
e anche in tutte le sue sottocartelle.

Load OpenApps

Questa istruzione avvia tutte le applicazioni che iniziano con "Theremino_" e finiscono con ".exe".

Le applicazioni vengono lanciate se si trovano nella cartella "Apps" e in tutte tutte le sue sottocartelle.

Load CloseApps

Dopo aver aperto molte applicazioni potrebbe essere necessario chiuderle. La funzione Load CloseApps chiude tutte le applicazioni che erano state aperte con "Load".

DA NOTARE
Quando si chiude Automation con la croce
tutte le applicazioni vengono automaticamente chiuse
(a meno che si stia lavorando sui sorgenti nel modo "Debug")

Load CloseAll

Questo comando chiude tutte le applicazioni come il comando precedente ma poi chiude *anche Theremino Automation.*

SUGGERIMENTO
Per evitare di chiudere Automation
tenere premuto CTRL mentre si esegue la istruzione "Load CloseAll"

DA NOTARE
Quando si chiude Automation con la croce
tutte le applicazioni vengono automaticamente chiuse
(a meno che si stia lavorando sui sorgenti nel modo "Debug")

Load CloseApps file-name

Per chiudere una singola applicazione dopo all'istruzione "Load CloseApps" si scrive anche il nome del file, come in questo esempio:

```
Load CloseApps C:\Theremino_HAL\Theremino_HAL.exe
```

Se non si specifica il percorso completo allora il percorso viene ricostruito a partire dlla posizione in cui si trova l'eseguibile "Theremino_Automation.exe", si può quindi scrivere il suo nome come nella linea seguente e se il file viene individuato allora il suo processo verrà chiuso.

```
Load CloseApps Theremino_HAL\Theremino_HAL.exe
```

Load CloseApps process-name

Per chiudere una applicazione si potrebbe anche scrivere il nome del "processo" che è il nome del file eseguibile della applicazione non preceduto dal percorso. Si scrive il nome con o senza la estensione exe finale, come negli esempi seguenti:

```
Load CloseApps Theremino_Editor.exe
Load CloseApps Theremino_Editor
```

Chiudere le applicazioni utilizzando il nome del processo può essere utile quando non si sa dove risiede l'eseguibile, come ad esempio nel caso di Theremino_Editor che non viene mai avviato direttamente, ma viene avviato dal sistema Windows quando si apre un file che è stato a lui associato.

Il nome del processo deve avere le lettere maiuscole e minuscole esatte.

Load CloseApps Windows

ATTENZIONE
Questo comando spegne il PC !

Questo esempio chiude il sistema operativo Windows:

```
Load CloseApps Windows
```

Load Slots / Save Slots

Con l'istruzione **Load Slots** si caricano tutti i valori degli Slot (dallo Slot 0 allo Slot 999), leggendoli dal file "_Slots.txt" (solo gli Slot numerici, non gli SlotText).

Il file "_Slots.txt" contiene i valori di tutti gli Slots, che erano stati preventivamente salvati con la istruzione **Save Slots**.

Con **Load Slots n1 n2** si caricano solo dallo Slot n1 allo Slot n2 compresi.

Al posto di n1 e n2 si possono anche scrivere variabili, oppure formule, anche complesse. Attenzione che le formule della istruzione Load Slots non devono contenere spazi, altrimenti danno un errore. Per utilizzare spazi in queste formule si potrebbe racchiuderle con doppi apici. Ecco alcuni esempi:

Load Slots 20 30	carica gli slot da 20 a 30 compresi
Load Slots v1 v2	carica gli slot da 10 a 20 compresi
Load Slots 2*3 v2+2	carica gli slot da 6 a 22 compresi
Load Slots 2 Math_PI*4	carica gli slot da 2 a 13 compresi

Negli esempi si suppone che v1 sia uguale a 10, e che v2 sia uguale a 20.

Nell'ultimo esempio si vede che PiGreco (funzione Math_PI che vale 3.14159...), viene moltiplicato per 4, e poi arrotondato all'intero più vicino (13).

Load Vars / Save Vars

Con l'istruzione **Save Vars** si salvano le variabili utilizzate nel file "_Vars.txt".

Con **Load Vars** si ricaricano le variabili che sono utilizzate e presenti nel file.

Se si utilizza **Save Vars** nell'evento **EventStop** e **Load Vars** nella parte iniziale del programma, si può ripristinare lo stato totale della applicazione che ripartirà esattamente come è stata chiusa.

ATTENZIONE

Quando si utilizza **Load Vars** bisogna fare caso
a dove si inizializzano le variabili e a come le si inizializzano.

Leggete la prossima pagina che spiega come utilizzare **Load Vars**

Load Vars - Attenzione alle inizializzazioni

L'esecuzione del comando **Load Vars** potrebbe ricoprire le vostre inizializzazioni creando comportamenti inattesi, strani e contro-intuitivi.

Per evitare problemi è bene posizionare l'istruzione Load Vars nelle prime righe della applicazione e scrivere subito dopo tutte le inizializzazioni.

Le inizializzazioni che non devono venire modificate devono stare nella zona iniziale, dopo **Load Vars** e prima di **Stop**.

```
' ------------------------------------
Load Vars
' ------------------------------------ Not affected by Load Vars
Array1 = 0.5,1,2,5,10,20,50,100,200
Array2 = 0,12,20,50,100,200,500,1000
InitLog
Numeric SoundFrequency = 1000
Numeric SoundMillisec = 10
' ------------------------------------ LOAD APPS
Load Theremino_SlotViewer.exe
' ------------------------------------ BUTTONS
Button 1 Label TestOnOff
Button 2 Label TestFrequency
' ------------------------------------ START ALL
OpenComPort
UpdateButtons
Loop
Stop

Label InitLog
  String LogFile = ".\LOGS\LOG_" + Format(Now, "yyyy_MM_dd_HH_mm") + ".txt"
  Numeric LogOldTime = -Infinity
Return
```

In questo esempio le zone evidenziate in giallo sono le inizializzazioni che non devono venire ricoperte da Load Vars.

Notare che sono valide anche le inizializzazioni posizionate nella funzione InitLog dato che la chiamata InitLog si trova nella zona valida.

In questo modo, se necessario, si potranno modificare le inizializzazioni e non verranno ricoperte da quello che viene ricaricato da Load Vars.

- - - - - -

Vedere anche gli esempi nella cartella "Demo Programs\Vars\"
L'esempio "SaveVars_Test.txt" fa il test di Save e Load con tutti i tipi di variabili.

Mentre l'esempio "SaveAndRestoreVars.txt" mostra un metodo più complesso che serve per salvare le variabili e ripristinarle utilizzando un file a piacere.

Load (Immagini, Video e Suoni)

Con l'istruzione LOAD si possono caricare file di immagini, video e suoni, che devono trovarsi nella cartella “Media”. Se si trovano in una sotto-cartella si deve aggiungere il suo nome, come in questo esempio: `Load "Sound\Ringer.mp3"`

I file di immagini possono avere le seguenti estensioni:
jpg jpeg bmp png exif tiff svg

I file video possono avere le seguenti estensioni:
avi wmv mpg mpeg m2ts m3u mp4 m4v mp4v 3g2 3gp2 3gp 3gpp mov gif

I file di suoni (e musica) possono avere le seguenti estensioni:
mp3 wav m4a aif aifc aiff au snd aac adt adts flac mid midi rmi
(se manca il file si sentirà un breve "beep" al suo posto)

Ecco qualche esempio di istruzioni che caricano questi file

```
Load Image1.jpg
Load Video1.avi
Load "Science Picnic 2014.mp3"
```

Prima di caricare i file audio, si può impostare il loro volume con la istruzione:
`Load Volume nn` (con nn da 0 a 100)
La istruzione `Volume` deve precedere la istruzione `Load` del file e ha effetto su tutti i file audio, tranne che sui file .mid e .midi

- - - - -

Dopo aver caricato i file video e audio, li si possono controllare con le istruzioni:
`Load Hide` rende invisibili le immagini e i video (riappare il programma).
`Load Stop` ferma l'esecuzione dei video e degli audio.
`Load Pause` mette in pausa l'esecuzione dei video e degli audio.
`Load Position Sec` modifica la posizione (in secondi) dei video e degli audio.
`Load Play` fa ripartire video e audio che erano in pausa.

- - - - - -

Vedere anche la pagina ***funzioni dei "Media"*** e gli esempi:
“Demo-ImagesVideoSounds.txt” nella cartella "Demo LOAD and MEDIA functions"

E anche i due esempi che mostrano nomi di file composti con stringhe e numeri:
“Demo – BM_Foundation1.txt” e “Demo - BM_Foundation2.txt”

Load (txt, pdf, doc, ecc...)

Con la istruzione Load invece di caricare e visualizzare file con automation, si può incaricare il sistema Windows di aprire ogni tipo di file. In questo caso Windows aprirà il file con il programma che è stato predefinito per la sua estensione.

Per cui solitamente se è un ".txt" verrà aperto con Notepad, se è un ".rtf" con WordPad, se è un ".doc", con Word o con OpenOffice e se è un ".pdf" con Google Chrome o con un PDF-Reader.

Attenzione
Non utilizzate **Load** per caricare un file da disco in una stringa!
Per fare questo si deve usare la funzione **GetTextFile**()
come in questo esempio `String str = GetTextFile("MyFile.txt")`

- - - - -

Se si fa Load di un file ".txt" che si trova nella cartella Programs. allora il file viene caricato come programma di automation. Nel caso invece sia in un'altra cartella, o inizi per "Files\", o abbia una estensione diversa da txt, allora viene caricato con il programma predefinito di Windows per la sua estensione.

Se il file non ha il percorso completo allora viene caricato dalla cartella "Files".

Per utilizzare un percorso relativo si deve tenere conto che la cartella di partenza è sempre "Programs". Nel seguente esempio si mostra come è possibile indicare che il file "Esempio1.pdf" si trova nella cartella "Files":

```
Load "..\Files\Esempio1.pdf"
```

In questo esempio i due punti iniziali significano: "salire di una cartella".

- - - - -

I file di immagini, video e musica (con estensioni .jpg .bmp .png .gif .avi .wmv .mpg .mp4 .mov .mp3 e .wav), vengono caricati da automation solo se si trovano nella cartella "Media".

Se invece non si trovano nella cartella "Media", allora vengono aperti dal sistema operativo Windows con il programma predefinito per la loro estensione.

Load (Program)

Con la istruzione Load si può caricare un programma di automation, scegliendolo tra quelli che sono presenti nelle cartella "Programs" di Automation.

Il nome del programma da caricare deve essere racchiuso tra doppi apici.

Il file del programma da caricare deve trovarsi nelle sottocartelle della applicazione Automation, a partire dalla cartella “Programs”.

Il file del programma da caricare deve terminare con la estensione “.txt”

Ecco un esempio:

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

Il programma che viene caricato sostituirà il programma attuale e la esecuzione continuerà immediatamente, partendo dalla prima linea del programma appena caricato.

Vedere gli esempi nella cartella
"Demo Programs \ Demo LOAD PROGRAM”

Load (Variabile da File)

Con la istruzione Load si può caricare un intero file di testo in una variabile, che può essere una di quelle predefinite (da s1 a s99) o una variabile "stringa" definita dall'utente.

Il nome del file da caricare nella variabile potrebbe essere racchiuso tra doppi apici, o anche senza apici. Il file deve avere estensione ".txt" e deve trovarsi nella stessa cartella del programma, oppure nella cartella "Files".

Ecco un esempio: `Load s1 SetupFile.txt`

Dopo aver caricato da file la variabile, la si può separare in linee e in campi, utilizzando le funzioni GetSeparatedStringCount e GetSeparatedString.

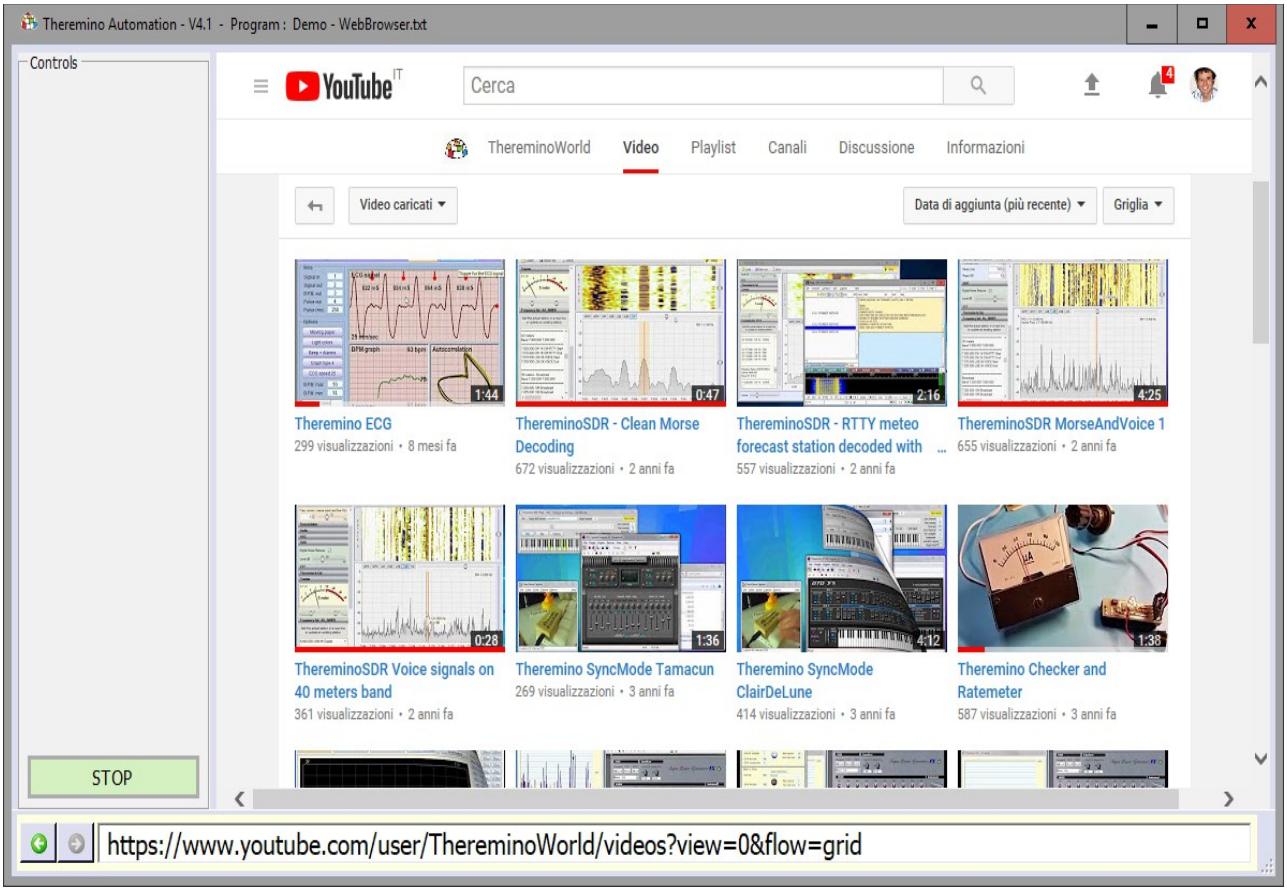
Vedere gli esempi nella cartella
"Demo Programs \ Demo LOAD TXT files"

Load (WEB address)

L'istruzione LOAD, seguita da un indirizzo WEB, apre una pagina Internet.

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

Quando si scrivono gli indirizzi nella istruzione LOAD, la parte iniziale http:// è necessaria.



In questa immagine si vede una pagina di YouTube, con i video del sistema theremino.

Una volta aperta una pagina è poi possibile navigare dovunque sul WEB, utilizzando i collegamenti che si trovano nelle pagine del sito.

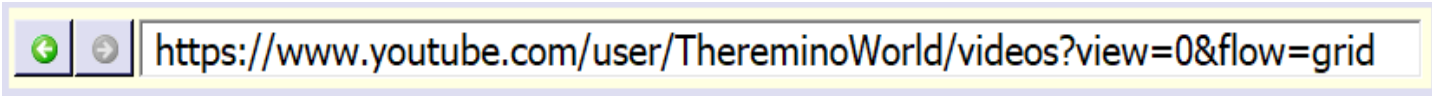
Oppure si può scrivere un nuovo indirizzo nella barra che si trova in basso, come spiegato nella prossima pagina.

Load (opzioni per le pagine WEB)

La barra degli indirizzi

Scrivendo un indirizzo in questa barra, si può visualizzare un altro sito WEB. Quando si scrive in questa barra, si può anche omettere lo http:// che precede l'indirizzo.

Dopo aver scritto l'indirizzo, per avviare la navigazione verso il nuovo sito, si deve premere ENTER.



Il due pulsanti sulla sinistra servono per aprire la pagina precedente o quella seguente.

Ingrandire e rimpicciolire la pagina

In alcuni casi potrebbe essere utile ingrandire la pagina, per leggere meglio i caratteri.

In altri casi potrebbe essere utile ridurre le sue dimensioni, per non dover utilizzare i cursori di scorrimento troppo spesso.

Per ingrandire o rimpicciolire la pagina WEB, si possono usare combinazioni di tasti, o la rotella del Mouse.

Prima di tutto si deve fare click sulla pagina web, per essere sicuri che sia selezionata, altrimenti i comandi seguenti non funzioneranno.

- Premere CTRL e il tasto “+”, per ingrandire la pagina.
- Premere CTRL e il tasto “-”, per ridurre le dimensioni della pagina.
- Premere CTRL e il tasto ZERO, per ripristinare l'ingrandimento normale.
- Premere CTRL e ruotare la rotella del mouse, per modificare l'ingrandimento.

Le istruzioni "Option"

Le istruzioni che iniziano con la parola "Option" modificano il funzionamento globale del programma. Perché abbiano effetto si deve eseguirle ad ogni avvio del programma, quindi solitamente le si scrivono nella zona iniziale, prima della istruzione "Stop". Si può comunque modificarle anche in seguito, durante l'esecuzione del programma.

Option GlobalKeys Enable

Normalmente l'istruzione Key e la funzione Key() agiscono solo se la applicazione Automation è selezionata e non è minimizzata. Ma se si esegue questa istruzione allora i tasti della tastiera agiscono sempre.

Se si abilita questa opzione si dovrà fare attenzione a non utilizzare la tastiera per altre applicazioni, altrimenti si potrebbero eseguire dei comandi per sbaglio.

Option GlobalKeys Disable

Dopo aver abilitato l'opzione GlobalKeys si potrebbe volerla disabilitare durante l'esecuzione del programma.

Questa istruzione ripristina lo stato normale con i tasti della tastiera che agiscono solo se l'applicazione Automation è selezionata e non è minimizzata.

Option Speed n

Normalmente si regola la velocità di esecuzione del programma con il cursore "Speed", spiegato in ***questa pagina***. Ma in alcuni casi, si potrebbe scrivere l'istruzione "Option Speed" nel programma stesso, e in tal caso anche il cursore Speed viene impostato dal programma.

Il numero "n" può variare da 1 a 9 ed è lo stesso numero indicato dal cursore Speed.

Option Transparency n

Invece di utilizzare il cursore "Transparency", spiegato in ***questa pagina***, si può regolare la trasparenza da programma, impostando un numero "n" da 0 a 9.

Vedere anche gli esempi nella cartella "Demo Programs \ Demo Option"

PressKeys

Questa istruzione invia caratteri e comandi, come se si premessero i tasti della tastiera o si premessero e rilasciassero i pulsanti del Mouse.

Tutte le lettere, numeri e comandi che si inviano con PressKeys devono essere separati con uno spazio, come in questo esempio:

```
PressKeys A B C D Enter
```

Per inviare spazi si utilizza il comando "Space", come in questo esempio:

```
PressKeys A B Space C D Enter
```

Gli spazi separano le lettere una dall'altra e permettono di inviare anche comandi.

Ecco la lista completa dei comandi e dei simboli:

```
Tab Esc Space Enter Return Left Right Up Down Backspace Pause
CapsLock PageUp PageDown End Home PrintScreen Insert Delete
ScrollLock NumMul NumAdd NumSub NumDiv NumLock NumDelete Num0 Num1
Num2 Num3 Num4 Num5 Num6 Num7 Num8 Num9 f1 f2 f3 f4 f5 f6 f7 f8 f9
f10 f11 f12 : ; = , < > _ . ? * - / ~ ` [ { \ | ] } ' 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 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 0 1 2 3 4 5 6 7 8 9 LButtonDOWN
LButtonUP MButtonDOWN MButtonUP RButtonDOWN RButtonUP
```

I comandi seguenti sono modificatori che vengono eseguiti prima di tutti gli altri caratteri. Tutti i caratteri verranno quindi eseguiti con il tasto modificatore premuto.

```
SHIFT CTRL ALT WINDOWS
```

Ecco due esempi che eseguono i tasti C e V tenendo premuto Control:

```
PressKeys CTRL C
PressKeys CTRL V
```

Tra un PressKeys e il successivo può essere necessario aggiungere una pausa (Wait) per dare tempo a Windows di eseguire il comando precedente.

Ecco un esempio che apre il menu Windows-x, e poi la finestra di comando e infine esegue un comando DIR.

```
PressKeys WINDOWS x
Wait Seconds 0.5
PressKeys i i Enter
Wait Seconds 0.5
PressKeys d i r Enter
```

Notare che l'esempio dipende dalla lingua italiana e dal menu contenente "Windows PowerShell". Notare anche che la x dopo a WINDOWS deve essere minuscola, altrimenti verrebbe inviata come SHIFT-X e il menu non si aprirebbe.

Il comando PressKeys può dipendere dalla lingua e dai tempi di esecuzione del sistema. Per maggiori informazioni leggere ***questa pagina*** e ***questa pagina***

PressKeys è usabile con una variabile che contiene i caratteri da inviare, ad esempio si può scrivere: `s1 = "d i r Enter"` e poi: `SendKeys s1`

SendKeys

Questa istruzione è simile alla precedente "PressKeys" ma ha alcuni vantaggi e anche degli svantaggi.

Il principale vantaggio è di poter scrivere il testo da inviare, senza separare le lettere con spazi. Ma in compenso i comandi vanno racchiusi con parentesi graffe, come in questo esempio: `SendKeys ABCD{ENTER}`

Vi sono anche altre sottili differenze, che rendono questo comando più comodo in alcuni casi, ma più scomodo in altri. Ad esempio con SendKeys non si riesce ad aprire il menu Windows-X dell'esempio della pagina precedente.

Anche i comandi sono leggermente diversi, ecco l'elenco completo:
`{TAB} {SHIFT} {CTRL} {ALT} {WINDOWS} {ESC} {ENTER} {LEFT} {RIGHT} {UP} {DOWN} {BACKSPACE} {BREAK} {CAPSLOCK} {PGUP} {PGDN} {END} {HOME} {PRTSC} {INSERT} {DELETE} {SCROLLLOCK} {NUMLOCK} {HELP} {ADD} {SUBTRACT} {MULTIPLY} {DIVIDE} {F1} {f2} {f3} {f4} {f5} {f6} {f7} {f8} {f9} {f10} {f11} {f12} : ; = , < > _ . ? * - / ~ ` [ { \ | ] } ' 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 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 0 1 2 3 4 5 6 7 8 9`

Per specificare che una combinazione di MAIUSC, CTRL e ALT deve essere mantenuta mentre vengono premuti diversi altri tasti, si racchiude il codice tra parentesi.

Ad esempio, per specificare di tenere premuto SHIFT mentre e e C sono premuti, si scrive "+(EC)". Per specificare di tenere premuto SHIFT mentre è premuto E, seguito da C senza SHIFT, si scrive "+EC".

Al posto dei tre caratteri `(+ ^ %)` si possono usare i modificatori `{SHIFT}{CTRL}{ALT}`, ma i tre caratteri `(+ ^ %)` restano comunque riservati come modificatori e non li si possono usare. Quindi per inviare un "+" si deve scrivere `{ADD}`

- - - - -

Per specificare le ripetizioni, utilizzare il formato {carattere numero}. È necessario inserire uno spazio tra il carattere (o comando) e il numero. Ad esempio, {LEFT 42} significa premere il tasto freccia sinistra 42 volte, oppure {H 10} significa premere H 10 volte.

- - - - -

Il comando SendKeys non è molto affidabile, può dipendere dalla lingua e dai tempi di esecuzione del sistema. Per ulteriori informazioni vedere ***questa pagina*.**

SendKeys è usabile con una variabile che contiene i caratteri da inviare, ad esempio si può scrivere: **`s1 = ABCD{ENTER}`** e poi: **`SendKeys s1`**

Print

Questa istruzione stampa, nella riga inferiore della applicazione, un valore numerico o una stringa di caratteri.

Dopo alla parola PRINT si può scrivere qualunque formula, anche complessa, composta con valori numerici e stringhe di caratteri.

Esempi:

```
S1 = "PI = "
S2 = Format(Math_PI, "")
```

```
Print S1 + S2
```
risultato: `PI = 3.14159265358979`

```
Print "Duck" = "Cat"
```
risultato: False

```
Print Mid("Duck", 4, 1) = "k"
```
risultato: True

```
Print "Donald Duck"
```
risultato: `Donald Duck`

```
Print CLS
```
risultato: Cancella tutto il testo

Di solito si utilizza questa istruzione per verificare il funzionamento del programma, per conoscere valori intermedi dei calcoli e per evidenziare che determinate righe vengano eseguite. *Per avere un area maggiore per il testo da stampare, si può usare l'istruzione* ***Controls OpenTextBox****.*

Inserire spazi nelle stringhe

A volte potrebbe essere utile inserire molti spazi in una stringa. Altre volte si potrebbero aggiungere spazi speciali che non permettono ai dispositivi di visualizzazione e stampa di andare a capo in quel punto.

Per inserire spazi si possono usare i metodi seguenti:

- Racchiudere la stringa con doppi apici e inserire gli spazi nella stringa.
- Utilizzare le funzioni PadLeft, PadRight e Chars.
- Utilizzare la costante NBSP (Non Breaking Spaces - Spazi non spezzabili)
- Inserire spazi di tipo NBSP nella stringa stessa con **ALT 255** (premere ALT e poi la sequenza di tasti 2 5 5 sul tastierino numerico, e infine rilasciare ALT).

- - - - - -

Per fare esperimenti con l'istruzione Print utilizzare gli esempi “Demo TextBox”

Save

Salvare gli Slot e le variabili

L'istruzione `Save Slots` salva i valori di tutti gli Slot (dallo 0 a 999), nel file "_Slots.txt" (istruzione valida solo per gli Slot numerici, non SlotText).

L'istruzione `Save Vars` salva i valori di tutte le variabili predefinite (v1..v99 e s1..s99), nel file "_Vars.txt"

Ulteriori informazioni nelle pagine ***Load/Save Slots*** e ***Load/Save Vars***

Appendere del testo a un file

L'istruzione `Save StringToFile s1 s2` appende il testo contenuto nella variabile s1, al file il cui nome è specificato da s2.

Il nome del file deve includere anche la estensione, solitamente "txt". Al posto di s1 e s2 si possono usare variabili definite dall'utente o anche scrivere direttamente delle stringhe di testo (racchiuse tra doppi apici).

Se manca il percorso allora il percorso diventa la cartella "Files", che si trova accanto al file "Automation.exe".

I file vengono normalmente scritti come Unicode (due byte per carattere) così da poter contenere anche i caratteri cinesi. Per scrivere un file in ASCII (un byte per ogni carattere) si deve anteporre ad ogni stringa che si scrive il testo "<ASCII>".

Attenzione anche a non "appendere" righe ASCII a un file pre-esistente di tipo Unicode, in tal caso cancellarlo preventivamente con l'istruzione DeleteFile.

Vedere: "Programs\Demo Programs\Files\WriteFile_ReadFile_SelectFile.txt"

Eliminare un file

L'istruzione `Save DeleteFile s1` cancella il file il cui nome è specificato da s1.

Il nome del file deve includere anche la estensione, solitamente "txt". Al posto di s1 si può usare una variabile definite dall'utente o anche scrivere direttamente una stringa di testo (racchiusa tra doppi apici).

Se il file non viene trovato allora il percorso diventa la cartella "Files", che si trova accanto al file "Automation.exe".

Copiare un file

Per copiare un file si utilizzano StringToFile e GetTextFile come in questo esempio:

```
Save StringToFile GetTextFile("File.txt") "FileCopy.txt"
```

Come sempre si possono usare variabili e se non si specifica il percorso allora viene usata la cartella "Files", che si trova accanto al file "Automation.exe".

Select - Case - CaseElse - EndSelect

Il costrutto Select-Case permette di scegliere tra una serie di casi. In questi esempi si vedono solo due “Case” ma potrebbero essere decine.

Ottenere lo stesso funzionamento con una serie di “IF” sarebbe più complesso e il funzionamento sarebbe meno evidente.

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

In questo esempio, se la variabile v1 vale 1, viene scelto il primo “Case”. Se vale 2, viene scelto il secondo “Case”.

Il “CaseElse” viene scelto se tutti i “Case” precedenti falliscono.

Questa implementazione dei Select-Case è particolarmente flessibile. Quasi tutti i linguaggi impongono che dopo ai “Case” ci sia un valore costante, invece in Automation si possono anche scrivere delle espressioni complesse.

Nella prossima immagine si vede un esempio semplice, solo funzioni che leggono Slot, addizioni e moltiplicazioni ma, sia dopo al “Select” che dopo ai “Case”, si possono scrivere espressioni di qualunque complessità.

```
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
```

Attenzione: alla fine ci vuole sempre un “EndSelect”

- - - - -

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo Select-Case”

Select - Case (caratteristiche speciali)

La nostra implementazione del Select-Case permette confronti inusuali, impossibili in quasi tutti i linguaggi di programmazione, ma molto utili.

Le istruzioni "Case" confrontano le stringhe senza tener conto delle lettere maiuscole e minuscole. Inoltre si può confrontare di tutto, stringhe, numeri, operatori booleani, ed espressioni con risultati veri o falsi, permutandoli a piacere, nelle istruzioni Select e Case.

Ad esempio si potrebbe scrivere “Select True”, e verrebbe eseguito il primo “Case” con una espressione che vale “True”.

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

Se invece si scrivesse “Select False”, verrebbe eseguito il primo “Case” con una espressione che vale “False”.

Oppure si potrebbe scrivere “Select Slot(1) = 3” che eseguirebbe il “Case True”, se lo Slot(1) vale 3. Altrimenti eseguirebbe il “Case False” (o il “CaseElse”).

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

Ma si potrebbe anche scrivere “Case Slot(2) = 5”, che si comporterebbe come un “Case True”, se Slot(2) vale 5. Altrimenti si comporterebbe come un “Case False”.

Si possono quindi fare ogni genere di affermazioni e di scelte, scrivendole nel modo che si preferisce, e che sembra più naturale.

- - - - -

Vedere, nella cartella "Demo Programs \ Demo Select-Case",
gli esempi “Demo – Select-Comparations”
e “Demo - Select-True-False”

Slot

Con questa istruzione si leggono e si scrivono gli Slot numerici.

Gli Slot sono il centro della comunicazione del sistema theremino e chi legge queste istruzioni **dovrebbe** già sapere cosa sono.

In caso contrario si consiglia di ***leggere questa sezione*** e la seguente sugli SlotText, e magari anche tutta la ***pagina sulle comunicazioni***, dall'inizio alla fine.

Ecco alcuni esempi che scrivono in uno Slot

```
Slot(1) = 12
```
Il valore 12 viene scritto nello Slot 1

```
Slot(2) = Rnd * 1000
```
Nello Slot 2 viene scritto un valore casuale da 0 a 1000

```
Slot(3) = Sin(1 / 3)
```
Nello Slot 3 viene scritto il numero 0.333333333333333

Questi esempi leggono da uno Slot

```
v1 = Slot(1)
```
Il valore dello Slot 1 viene letto e assegnato alla variabile v1

```
If Slot(2) < 500
     Beep
EndIf
```
Se lo Slot 2 è minore di 500 viene emesso un suono

Sono possibili anche espressioni più complesse, la seguente espressione scrive il valore 16 nello Slot 3.

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

- - - - -

Vedere anche gli esempi nella cartella “Demo Programs \ Demo Slots”

Per visualizzare i valori delle variabili e degli Slot,
si consiglia di aprire la finestra di Debug, con il tasto destro del Mouse.

Lo Slot dei comandi

Per comunicare con le applicazioni HAL si utilizza uno Slot speciale, lo "Slot dei Comandi".

Attraverso lo Slot dei comandi (per mezzo di numeri speciali chiamati NAN) si inviano istruzioni alle applicazioni HAL e si possono anche leggere informazioni sul numero dei moduli connessi e sugli errori, ecco alcuni esempi:

`Slot 0 = Recognize` Le applicazioni HAL devono riconoscere i moduli collegati (come se si premesse il pulsante "Riconosci" sull'HAL).

`Slot 0 = Calibrate` Le applicazioni HAL devono calibrare tutti i moduli che possono essere calibrati, solitamente i CapSensor (come se si premesse il pulsante "Calibra" sull'HAL).

Esistono anche dei comandi che non si inviano allo Slot dei comandi, ma direttamente agli Slot cui sono collegati gli attuatori, come ad esempio Servo-motori o motori Stepper. Ecco alcuni esempi:

`Slot 1 = Sleep` Toglie alimentazione al Servo-motore collegato allo Slot 1

`Slot 3 = Reset` Il motore Stepper collegato allo Slot 3 viene reimpostato con un nuovo valore (per i particolari vedere le istruzioni degli HAL).

Dallo Slot dei comandi si possono anche leggere informazioni riguardo ai dispositivi collegati. Dopo un comando "Riconosci" o dopo l'avvio della applicazione HAL lo Slot dei comandi contiene il numero di dispositivi collegati e lo si può leggere ad esempio con:

`v1 = Slot(0)` In v1 si ottiene il numero dei dispositivi (oppure -1 durante il riconoscimento).

Lo Slot dei comandi può contenere numeri NAN corrispondenti alle stringhe:
"Sleep", "Reset", "Recognize", "Calibrate", "Master disconnected ERROR", "No Masters"

`S1 = DecodeCommandSlot(0)` Questa funzione legge comandi e errori come stringhe.

- - - - -

Normalmente lo Slot dei comandi è lo Slot Zero ma, quando si usassero più applicazioni HAL sullo stesso PC, si potrebbe utilizzare uno Slot diverso per ognuna di esse. In questi casi si apre il file INI della applicazione HAL con il Notepad e si modifica la riga "CommandSlot= 0" con un numero diverso da zero.

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo SlotZeroCommands”

SlotText

Questa istruzione scrive e legge stringhe di testo negli SlotText.

Gli Slot sono il centro della comunicazione del sistema theremino e chi legge queste istruzioni **dovrebbe** già sapere cosa sono.

In caso contrario si consiglia di ***leggere questa sezione*** e la seguente sugli SlotText, e magari anche tutta la ***pagina sulle comunicazioni***, dall'inizio alla fine.

Attenzione a non confondere gli Slot con gli SlotText, hanno indirizzi simili (da 0 a 999), ma scrivono e leggono in zone di memoria diverse.

Inoltre gli Slot contengono numeri (interi o a virgola mobile), mentre gli SlotText contengono stringhe di caratteri (fino a 100000 caratteri).

Con gli SlotText si possono comandare varie applicazioni, ad esempio: HAL, ArduHAL, IotHAL, QRdecoder, Cobot, Blockly, CNC, Dictation, GPS,Loggers, Motors, OpenAi, Player, SloTextToSpeech, Spectrometer e SlotViewer.

Nella cartella "Programs\Demo Programs\SlotText Commands" ci sono molti esempi per provare a comandare varie applicazioni con gli SlotText.

- - - - -

Ecco alcuni esempi che scrivono stringhe di caratteri in uno SlotText

```
SlotText(1) = "Casa " + "sull'albero"
```
Il testo viene scritto nello SlotText 1

```
SlotText(2) = Rnd
```
Viene scritto un valore casuale trasformato in testo

```
SlotText(3) = 2.5 + 3
```
Nello SlotText 3 viene scritto il valore 5.5 trasformato in testo

Questi esempi leggono stringhe di caratteri da uno SlotText

```
s1 = SlotText(1)
```
Una stringa viene letta dallo SlotText e assegnata alla variabile s1

```
If SlotText(2) = "OK"
     Beep
EndIf
```
Se lo SlotText 2 contiene "OK" viene emesso un suono

Speed

La istruzione "Speed n" è diventata "Option Speed n". Vedere ***questa pagina***.

Stop, End

Con STOP il programma si ferma, ma continua a rispondere alla tastiera del PC (istruzione **Key n Goto/Gosub xx**), ai pulsanti (istruzione **Button n Text xx**), e anche ad eventuali Slot (istruzione **Button n Slot nn)**.

Invece END termina totalmente l'esecuzione del programma. Per farlo ripartire si dovrà premere il pulsante RUN.

TTS (Text to speech)

Queste istruzioni traducono il testo in voce. In Windows 10 sono solitamente presenti almeno la propria lingua e l'inglese. Per aggiungere altre lingue si utilizza "Impostazioni" , "Data ora e lingua" , "Lingua" e infine con "Aggiungi una lingua".

TTS SelectVoice "xxxx" Seleziona una voce tra quelle installate
Esempi:"ENG","ENGMALE","ENGM","ENGF","ENGN","ENG Neutral","FRA", "ITA FEM.","CHI","CHI F","ChiMale","ZHO M","ZHO","CHI neutral"

TTS SetVolume n	Regolazione del volume (valori da 0 a 100)
TTS SetSpeed n	Regolazione della velocità (valori da -10 a 10)
TTS Speak "xxx"	Pronuncia il testo "xxx"
TTS SpeakWait "xxx"	Pronuncia il testo "xxx" e aspetta la fine
TTS Stop	Smetti immediatamente di parlare
TTS Pause	Interrompi temporaneamente la pronuncia
TTS Resume	Riprendi a pronunciare il testo in pausa

TTSvoices	Risponde la lista delle voci installate
TTSlanguage	Risponde il nome "ISO" (ENG, CHI..) della voce
TTSready	Risponde "True" se TTS è inattivo
TTSpaused	Risponde "True" se TTS è in pausa
TTSspeaking	Risponde "True" se TTS sta parlando

Impostando il Volume a zero la funzione TTS Speak non agisce e TTSready risponde immediatamente "True".

Per pronunciare numeri e stringhe esistono anche le applicazioni ***SlotsToSpeech*** e ***SlotTextToSpeech***. E c'è anche la applicazione ***Voice*** che, oltre a parlare, riconosce le parole tra un elenco di parole e frasi. Queste applicazioni funzionano anche da sole, ma si potrebbe farle interagire con Automation attraverso gli Slot.

Vedere anche gli esempi nella cartella “ Demo Programs \ Demo TTS ”

Variable

Il linguaggio Automation contiene alcune variabili predefinite.

```
v1 v2 v3 ... v98 v99
```
<- Variabili numeriche predefinite

```
s1 s2 s3 ... s98 s99
```
<- Variabili stringa predefinite

Oltre alle variabili predefinite si possono definire altre variabili con nomi a piacere

```
Variable Numeric Pluto
```
<- Variabile numerica definita dall'utente

```
Variable String str1
```
<- Variabile stringa definita dall'utente

Si possono definire le variabili anche omettendo la parola "Variable"

```
Numeric Pluto
```
<- Variabile numerica definita dall'utente

```
String str1
```
<- Variabile stringa definita dall'utente

Tutte le variabili devono iniziare con una lettera (minuscola o maiuscola)
ed essere lunghe almeno due caratteri.

Il linguaggio Automation deve essere semplice da capire per cui non esistono variabili locali. Tutte le variabili, non importa dove e come siano definite, hanno lo stesso valore in ogni punto e in ogni funzione del programma.

Fate quindi molta attenzione a usare in ogni funzione variabili diverse
(a meno che vogliate utilizzarle per comunicare valori tra una funzione e l'altra)

Tutte le variabili non inizializzate contengono il valore zero o una stringa vuota. Per dare un valore alle variabili si scrive come in questi esempi:

```
v1 = 12
s1 = "Pluto"
value1 = 123
str1 = "This is a string"
```

Queste righe agiscono solo se vengono eseguite.
Altrimenti le variabili rimangono non inizializzate.

Variable - Assegnazioni immediate

Quando si definiscono nuove variabili si può anche assegnare ad esse un valore immediato, come in questi esempi:

```
Variable Numeric Pluto = 12 * 44
Variable String str1 = "This is a string" + Str(Pluto)
```

I valori immediati vengono assegnati a tutte le variabili **nel momento in cui si avvia il programma** (con RUN), **anche se le righe di assegnazione non vengono eseguite**.

Se in seguito il programma esegue queste righe allora i valori vengono nuovamente assegnati, eventualmente anche con valori differenti.

Se le assegnazioni contengono formule e altre variabili (come il "`+ Str(Pluto)`" di questo esempio), allora l'effettivo valore che si assegna alla variabile dipende da cosa c'era in "Pluto" in precedenza.

Fate attenzione a questi comportamenti perché in alcuni casi quello che avviene potrebbe essere poco intuitivo.

Ad esempio, se durante la assegnazione dei valori si utilizzano variabili (come la variabile "Pluto" dell'esempio di questa pagina) queste all'avvio del programma potrebbero essere già inizializzate, oppure potrebbero non esserlo.

In altre parole il valore delle variabili può dipendere dall'ordine con cui si scrivono le assegnazioni, e in seguito anche dall'ordine con cui verranno eseguite sia le righe delle assegnazioni immediate, che quelle che contengono assegnazioni normali.

Il tutto si complica se le assegnazioni contengono formule e rifermenti ad altre variabili che a loro volta potrebbero dipendere da altre variabili ancora...

Nel dubbio scrivete una funzione di inizializzazione e chiamatela prima di utilizzare le variabili. Qualcosa di simile a questo esempio, ma da studiare volta per volta con attenzione:

```
Label InitAllVariables
     String str1 = "This is a string"
     Numeric Pluto = 12 * 44
     String str2 = str1 + Str(Pluto)
Return
```

Sperimentate con gli esempi nella cartella
“Demo Programs \ Demo Immediate Vars”

VarsFromFile

In alcuni casi potrebbe essere utile caricare i valori delle variabili da un file di configurazione. Così si potrebbe, ad esempio, far lavorare lo stesso programma su macchine diverse, con diverse dimensioni e diversi limiti per i movimenti.

Questo comando potrebbe essere utile anche in altre occasioni, perché è un comando generico, utilizzabile anche più volte nello stesso programma e anche leggendo file diversi in vari punti del programma.

Funzionamento del comando VarsFromFile

Ogni volta che si esegue il comando VarsFromFile NomeFile.txt tutte le variabili indicate nel file vengono aggiornate con i valori definiti nel file.

Il file deve esistere e non deve contenere errori, altrimenti appare un messaggio di errore e il programma si ferma.

Nel file le variabili si scrivono una per riga, seguite da un uguale e dal valore da assegnare alla variabile, come nell'esempio seguente.

```
Slot_LedR    =    907
Slot_LedG    =    908
Slot_LedB    =    909
Array1       =    11 22 33
Array2       =    11,22,33
```

Per evitare errori tutte le variabili indicate nel file devono essere dichiarate nel programma in esecuzione.

I nomi delle variabili devono corrispondere, ma non importa se i caratteri sono scritti minuscoli o maiuscoli e prima e dopo l'uguale si possono utilizzare spazi o tabulazioni a piacere.

Nel caso di variabili stringa è possibile includere gli spazi nella stringa racchiudendola tra due doppi apici. come mostrato in questi esempi:

```
StringVar1   = "   ABCD   " ' stringa con spazi
StringVar2   =     ABCD     ' spazi iniziali e finali rimossi
```

Si può posizionare il file nella cartella "Files" o nella cartella principale della applicazione (che contiene Automation.exe). Si può anche posizionarlo in sotto-cartelle, ma in questo caso si deve premettere al nome del file la parte di percorso necessaria per scendere nella sotto-cartella.

Sperimentate con l'esempio "VarsFromFile”
che si trova nella cartella “Demo Programs \ Demo Vars"
L' esempio "VarsFromFile" legge le variabili
dal file "VFF_Example.txt" che si trova nella cartella "Files"

Wait

Il programma si ferma sulla linea della istruzione WAIT, e proseguirà dalla linea seguente, al verificarsi di alcune condizioni.

Ecco alcuni esempi:

`Wait Seconds 1.53`	Attende un secondo e 53 centesimi
`Wait Button "Button 1"`	Attende la pressione del Button 1 (se inizializzato)
`Wait Slot(4) > 500`	Attende che il valore dello Slot 4 diventi maggiore di 500
`Wait AppRunning("xx.exe")`	Attende fino a che la applicazione è in esecuzione
`Wait Not AppRunning("xx.exe")`	Attende fino a che la applicazione è terminata
`Wait Application xx.exe`	(vecchio metodo) Attende applicazione chiusa

Se si specifica solo il nome del file della applicazione
(ad esempio **"Theremino_SlotViewer.exe"**)
allora la applicazione viene cercata nella cartella **"Apps"**
e anche in tutte le sue sottocartelle.

Vedere anche gli esempi nella cartella “Demo Programs \ Demo Wait”

Window (comandi)

```
Window MinSize
Window Sizable
Window Maximized
Window FullScreen
```

Con questi comandi si possono modificare le dimensioni della finestra, durante l'esecuzione del programma.

Le immagini della prossima pagina mostrano come appare la finestra del programma nelle varie dimensioni.

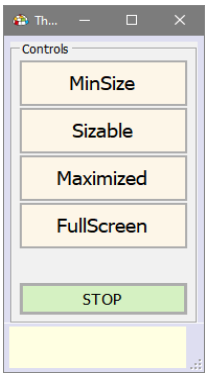
```
Window SetFocus
```

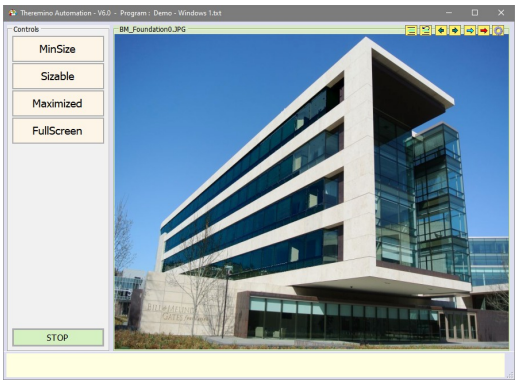
Con l'istruzione "SetFocus" si attiva la finestra di Automation e le si da il "focus" del sistema Windows.

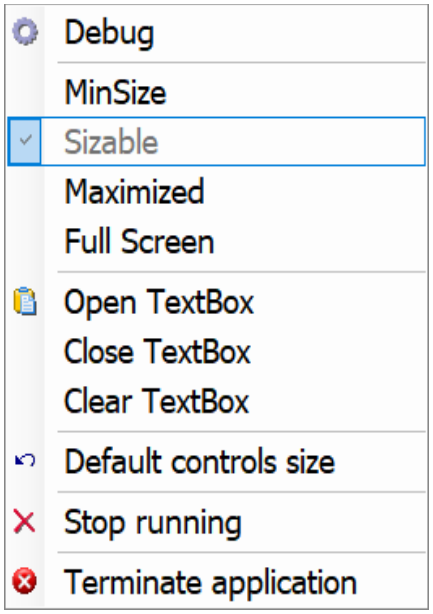
Questo comando può essere utile quando si vuole avere la certezza di ricevere gli eventi dei tasti premuti sulla tastiera.

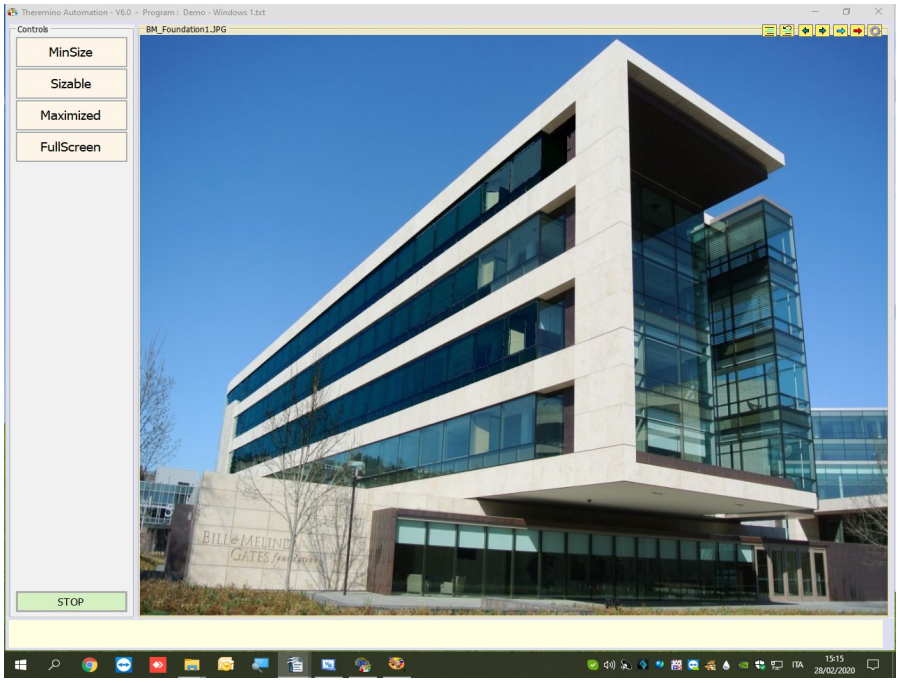
Window

Si possono anche scegliere manualmente le opzioni di dimensione della finestra, utilizzando i comandi del ***menu della applicazione***, che si apre facendo click con il pulsante destro del Mouse sulla parte sinistra della applicazione.









Vedere anche l'esempio
"Demo - Windows.txt"

Chiamare le funzioni

Eliminare i Gosub e passare i parametri alle funzioni semplifica i programmi e rende più leggibile il codice.

La funzione cui si passano i parametri inizia, ad esempio, così:

```
Label Function1
     Variable Numeric Param1
     Variable Numeric Param2
     Variable String  Param3
     ....
```

Se nelle versioni precedenti di Automation si scriveva qualcosa del genere:

```
Param1 = 12
Param2 = Rnd
Param3 = Now
Gosub Function1
Param1 = 567
Param2 = 89
Param3 = "23/07/2022"
Gosub Function1
Param1 = 10
Param2 = 11
Param3 = Now
Gosub Function1
```

Ora la stessa sequenza si semplifica con:

```
Function1(12)(Rnd)(Now)
Function1(567)(89)("23/07/2022")
Function1(10)(11)(Now)
```

- - - - -

I parametri delle funzioni si scrivono tra parentesi tonde.

I parametri di tipo stringa devono essere racchiusi tra doppi apici.

I particolari del passaggio dei parametri
sono spiegati nella prossima pagina.

I parametri delle funzioni

La funzione cui si passano i parametri deve iniziare con una o più righe di variabili, che devono stare tutte prima di ogni altra riga contenente istruzioni.

Le variabili dei parametri possono essere in qualunque numero e possono essere tutte numeriche o tutte stringa o mischiate in qualunque modo.

Attenzione che a differenza di altri linguaggi le variabili NON sono locali alla funzione in cui sono scritte, ma sono leggibili e scrivibili da tutte le altre zone del programma. Questo costringe a scrivere molte più variabili, tutte con nomi diversi, ma facilita la programmazione per chi ha poca esperienza.

Un buon modo per generare variabili sicuramente diverse è usare le iniziali del nome della funzione come nell'esempio seguente.

```
Label FunctionWithParams
     Variable Numeric FWP1 '<-- FWP iniziali del nome
     Variable Numeric FWP2
     Variable String  FWP3
     ....
```

Quando si chiama la funzione si scrivono i parametri tra parentesi tonde come negli esempi seguenti (per ogni parametro gli spazi e i TAB iniziali e finali non contano).

```
FunctionWithParams(Rnd)( -77.234)( "The quick brown fox")

FunctionWithParams( Sin(x) )( Cos(x) )("The day is: " + Now)
```

I parametri di tipo stringa devono essere racchiusi tra doppi apici, come nei due esempi precedenti, altrimenti si otterranno degli errori.

Ogni parametro sarà disponibile nella variabile corrispondente. Se si scrivono più parametri tra parentesi di quante sono le variabili, allora i parametri in eccesso non avranno effetto. E se invece ci sono più variabili dei parametri tra parentesi, allora le variabili in eccesso non verranno impostate.

Se si vuole lasciare invariata una variabile e scrivere quelle seguenti si utilizza la parola `(Null)` come nell'esempio seguente.

```
FunctionWithParams(Sin(x))(Null)("The day is: " + Now)
```

Vedere anche gli esempi nella cartella
"Demo Programs \ SlotText Commands"

Espressioni e funzioni

Invece di numeri o stringhe di caratteri, si possono anche utilizzare espressioni complesse. Le espressioni possono contenere funzioni, operatori, variabili e costanti. Gli IF e i Select-Case, possono anche contenere condizioni booleane.

Le funzioni

```
Slot(n) Sin(v) Cos(v) Asin(v) Acos(v) Tan(v) Atan(v) Atan2(v, v)
Sqrt(v) Abs(v) Sign(v) Pow(v, v) Exp(v) Exp2(v) Exp10(v) Log(v)
Log2(v) Log10(v) Int(v) Round(v) Mid(str, index) Mid(str, index,
len) Len(str) Trim(str) PadLeft(str, places) PadRight(str, places)
UCase(str) LCase(str) WCase(str) Format(v, style) Format(v) Str(v,
style) Str(v) Chars(str, number) Replace(str, oldStr, newStr)
RemoveComments(str) Chr(n) Asc(str) Bin(n) FromBin(str) Hex(n)
FromHex(str) Now() Today() Date(year, month, day) ElapsedTime Rnd
Rad(v) ConvertToIEEE754(Value, ConvertSingle, ReverseBytes)
COM_Status COM_Received COM_Portnames MouseX MouseY MouseXP
MouseYP MouseButtons GetSeparatedStringCount(str) CommandText()
GetSeparatedStringCount(str, separator) GetSeparatedString(str)
GetSeparatedString(str, separator) Key(str) DecodeXML(str)
FormatXML(str) FormatXML(str,n) Limit(n, Min, Max) LimitReached
```

Gli operatori

```
And Or Xor + - * /
```
And, Or e Xor sono booleani e anche numerici

Le variabili definite dall'utente (esempi)

```
Variable Numeric Pluto
```
Variabile numerica definita dall'utente

```
Variable String  str1
```
Variabile stringa definita dall'utente

Le variabili predefinite

```
v1 v2 v3 ... v97 v98 v99
```
Variabili numeriche predefinite

```
s1 s2 s3 ... s97 s98 s99
```
Variabili stringa predefinite

Le costanti

```
False True CRLF CR LF NBSP TAB Math_PI Math_E Infinity
```
Infinity = 9e99

Le condizioni

```
Le condizioni sono espressioni che terminano con Vero o Falso.
Vero o Falso è determinato da confronti (> < = >= <= <>).
```

I valori numerici

```
Il separatore dei decimali è sempre il punto. Esempio: 3.1416

Si può anche utilizzare la notazione esponenziale.
Ad esempio: 3e5  che significa 3 con cinque zeri, cioè 300000
oppure:  -2.5e2  che sta per -250
o anche:  7.3e-5 che sta per 0.000073
```

Risultati delle espressioni

Questo linguaggio converte automaticamente Booleani, Numeri e Stringhe a seconda delle necessità, per cui in alcuni casi lo ZERO potrebbe essere considerato False, e ogni altro numero considerato True.

Ecco alcuni esempi di espressioni e condizioni

`2 * 3 + 4 * Sqrt(16)`	Questa è una espressione, con risultato 22
`PI * 2`	Questa è una espressione, con risultato 6.2831853....
`"Duck" = "Cat"`	Questa è una condizione, con risultato False

Per altri esempi di funzioni aprire il file “Demo - Print.txt”
e quelli della cartella "Demo Functions and Errors"

La funzione Slot()

Questa funzione legge un valore numerico da uno Slot del sistema Theremino.

Ad esempio la seguente riga legge il valore dello Slot 2 e lo mette nella variabile v1:

```
v1 = Slot(2)
```

Esiste anche una espressione per scrivere negli Slot, ad esempio la riga seguente scrive un valore numerico nello Slot 12:

```
Slot(12) = 500.33
```

Si possono comporre anche espressioni complesse e usare variabili come indici degli Slot. Ad esempio le tre righe seguenti copiano il contenuto degli Slot da 0 a 9 negli Slot da 100 a 109, incrementando di 500 i valori numerici.

```
For v1 = 0 To 9
        Slot(v1 + 100) = Slot(v1) + 500
Next
```

Esistono anche istruzioni speciali per lo Slot dei comandi (solitamente lo Slot Zero). Vedere la pagina ***Slot Zero - Comandi Speciali***. Ecco alcuni esempi:

```
S1 = DecodeCommandSlot(0)
Slot(0) = Sleep
```

Le funzioni numeriche

Fate attenzione che tutte le funzioni in Automation hanno le parentesi, non usate quindi % per il modulo o ^ per elevare a potenza.

Pi - Ritorna il valore di Pi Greco (3,14159265358979)

Sin (numero) - Ritorna il seno del numero

Cos (numero) - Ritorna il coseno del numero

Asin (numero) - Ritorna l'arcoseno del numero

Acos (numero) - Ritorna l'arcocoseno del numero

Tan (numero) - Ritorna la tangente del numero

Atan (numero) - Ritorna la arcotangente del numero

Atan2 (numero, numero) - Arcotangente estesa ai quattro quadranti

Sqrt (numero) - Ritorna la radice quadrata del numero

Abs (numero) - Ritorna il valore assoluto del numero (sempre positivo)

Sign (numero) - Ritorna il segno del numero (-1, zero o +1)

Min (numero, numero) - Ritorna il minore tra i due numeri

Max (numero, numero) - Ritorna il maggiore tra i due numeri

Mod (numero, modulo) - Ritorna il primo numero limitato dal modulo

Int (numero) - Ritorna il numero intero, arrotondato in basso

Rad (numero) - Ritorna il numero convertito in radianti (n * PI / 180)

Round (numero) - Ritorna il numero arrotondato all'intero più vicino

Rnd - Ritorna un numero casuale tra zero (compreso), e uno (non compreso). Il numero è totalmente casuale perché viene usata la funzione Randomize.

Pow (numero, numero) - Ritorna il primo numero elevato al secondo

Exp (numero) - Restituisce la base del logaritmo naturale, elevata a "numero"

Exp2 (numero) - Restituisce "2" elevato a "numero"

Exp10 (numero) - Restituisce "10" elevato a "numero"

Log (numero) - Restituisce il logaritmo naturale (in base "e") del "numero"

Log2 (numero) - Restituisce il logaritmo in base "2" del "numero"

Log10 (numero) - Restituisce il logaritmo in base "10" del "numero"

Limit (numero, Min, Max) - Restituisce "numero" limitato tra Min e Max

LimitReached - Restituisce True se la funzione Limit ha limitato un "numero"

Vedere anche gli esempi nella cartella "Demo Programs \ Demo Math Functions"

Le funzioni per le stringhe

Left (stringa, n) - Ritorna i primi "n" caratteri della stringa
Right (stringa, n) - Ritorna gli ultimi "n" caratteri della stringa
PadLeft (stringa, len) - Ritorna la stringa addizionata di spazi fino a "length"
PadRight (stringa, len) - Ritorna la stringa addizionata di spazi fino a "length"

Len (stringa) - Ritorna la lunghezza della stringa (numero di caratteri)
Trim (stringa) - Ritorna la stringa senza spazi e TAB, iniziali e finali

Ucase (stringa) - Ritorna la stringa con tutti i caratteri maiuscoli
Lcase (stringa) - Ritorna la stringa con tutti i caratteri minuscoli
Wcase (stringa) - La prima lettera maiuscola e le altre minuscole

Chars (string, number) - Ritorna la stringa "string" ripetuta "n" volte

Replace (stringa, s1, s2) - Ritorna una stringa con tutte le occorrenze della stringa s1 rimpiazzate con la stringa s2.

RemoveComments(string) - Ritorna una stringa senza la parte finale contenente i commenti. I commenti iniziano con singolo apice ('), che viene considerato solo se si trova fuori da una stringa, cioè non racchiuso tra doppi apici.

Mid (stringa, start, len) - Ritorna una stringa a partire da “start” e lunga “len”
Mid (stringa, start) - Ritorna una stringa a partire da “start” e fino alla fine
Il parametro "start" delle funzioni Mid() inizia da "1". Se start vale "0" o un numero minore di zero, la funzione ritorna sempre una stringa vuota.

GetSeparatedStringCount (string, separatore opzionale = spazio)
GetSeparatedString (stringa, indice, separatore opzionale = spazio)
Queste due funzioni contano e estraggono sotto-stringhe da una stringa spezzandola in corrispondenza di un separatore, che può essere un singolo carattere o una stringa di caratteri. Se si omette il separatore allora viene utilizzato uno spazio. Ripetizioni del separatore contano come un separatore singolo.
La GetSeparatedString risponde la sotto-stringa indicata dal numero "indice". La prima sotto-stringa ha indice zero.

StringContains (string, "testo-da-cercare") - Risponde "TRUE" se la stringa "testo da cercare" è contenuta nella stringa "string". Questa funzione considera le lettere maiuscole e minuscole come equivalenti.

Le funzioni di conversione

Chr (numero) - Ritorna il carattere corrispondente al numero (da 0 a 65535)

Asc (stringa) - Ritorna il valore ASCII del primo carattere della stringa

Bin (numero) - Ritorna una stringa con il numero convertito in binario
Hex (numero) - Ritorna una stringa con il numero convertito in esadecimale

FromBin (stringa) - Ritorna il numero corrispondente alla stringa binaria
FromHex (stringa) - Ritorna il numero corrispondente alla stringa esadecimale

Val (stringa) - Trasforma in numero un valore numerico contenuto in una stringa

ConvertToIEEE754(Numero, ConvertSingle, ReverseBytes)
Ritorna una stringa che contiene quattro o otto valori esadecimali, con il numero convertito secondo il formato IEEE754.
Se in ConvertSingle si scrive True vengono restituiti quattro valori altrimenti otto.
Se in ReverseBytes si scrive True i bute vengono invertiti nel formato LittleEndian.
Esempio : ConvertToIEEE754(1.234, True, True) = "B6 F3 9D 3F"

- - - -

Le prossime quattro funzioni sono simili tra loro, convertono un numero in stringa utilizzando molte opzioni e formati.

Format (numero, stile) - Converte numero in stringa
Format (numero) - Converte numero in stringa

Str (numero, stile) - Converte numero in stringa
Str (numero) - Converte numero in stringa

Per i particolari leggere la pagina seguente

Le funzioni Format() e Str()

La funzione Format (numero, stile) converte un numero in stringa.

Nel parametro “stile” si deve fornire una stringa che specifica come deve apparire il numero convertito.

`Format(1234.56, "000000.000")`	Risultato: "001234.560"
`Format(1234.56, "0.0")`	Risultato: "1234.6"
`Format(1234.56, "0")`	Risultato: "1235"
`Format(Math_PI, "")`	Risultato: "3.14159265358979"
`Format(1234.56, "+0.000;-0.000")`	Risultato: "+1234.6" o "-1234.6"

Se si preferisce un nome più corto, si può usare la funzione Str (numero, stile) che è perfettamente identica alla funzione Format (numero, stile)

`Str(1234.56, "000000.000")`	Risultato: "001234.560"
`Str(1234.56, "0.0")`	Risultato: "1234.6"

Ambedue le funzioni sono usabili anche omettendo il parametro dello stile. In questo caso il numero verrà convertito con la massima precisione.

`Format(Math_PI)`	Risultato: "3.14159265358979"
`Str(Math_PI)`	Risultato: "3.14159265358979"

Sono possibili molte altre opzioni di stile, ad esempio i prossimi due stili convertono in notazione scientifica con una o due cifre di esponente.

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

e la notazione seguente converte l'istante presente in una data

`Str(Now, "yyyy/MM/dd HH:mm:ss")`	Risultato: 2024/11/30 09:50:14

Per una spiegazione completa di tutti i formati possibili leggere ***questa pagina***.

In questa applicazione (e in tutto il mondo scientifico),
per i decimali si usa sempre il punto,
anche nelle nazioni e nei sistemi operativi che usano la virgola.

Le funzioni MouseX, Y, XP e YP

MouseX e MouseY restituiscono la posizione del cursore del mouse sullo schermo normalizzata da zero (a sinistra o in basso) fino a uno (a destra o in alto). I valori normalizzati sono utili per effettuare regolazioni come se si disponesse di un Joystick o di due potenziometri.

In caso di schermi multipli questa funzione utilizza solo lo schermo su cui è posizionata la finestra di Automation.

MouseXP e MouseYP restituiscono la posizione del cursore del mouse sullo schermo con valori in pixel. Per cui a sinistra in basso i due valori valgono zero, mentre a destra in alto i due valori dipendono dal numero di pixel dello schermo.

In caso di schermi multipli i pixel iniziano da in basso a sinistra del primo schermo fino a in alto a destra dell'ultimo schermo e se lo schermo principale non è il primo allora i numeri che identificano i pixel possono anche essere negativi.

La funzione MouseButtons

MouseButtons risponde un numero che indica il pulsante del mouse premuto.

1 = pulsante sinistro / 2 = pulsante destro / 4 = pulsante centrale

Pulsanti premuti insieme producono numeri che sono le somme dei singoli pulsanti. Per cui i numeri prodotti da questa funzione possono andare da 0 (nessun pulsante premuto), fino a 7 (tutti e tre i pulsanti premuti contemporaneamente).

La funzione MouseWheel

Questa funzione fornisce un numero negativo o positivo che dice di quanti scatti è stata ruotata la rotella dall'ultima volta che la si è letta.

Per usarla, ad esempio per nuovere un asse di un braccio robotico si dovrebbe:

- Leggere MouseWheel frequentemente e sommarlo al valore precedente.
- Limitare il valore tra min e max di quell'asse.
- Dare il valore limitato all'asse.

Non è necessario che la applicazione Automation sia selezionata, ma il cursore deve essere sopra al riquadro "Controls" che contiene i "Button". Questo serve per evitare di muovere il motore per sbaglio, se si usa la rotella su altre applicazioni.

Guardate l'esempio Test_MouseWheel.txt
nella cartella Programs\Demo Programs\Mouse

Le funzioni che leggono i colori dei pixel

Queste due funzioni leggono il colore dei pixel dello schermo.

GetCursorPixelColor Restituisce il colore del pixel puntato dal cursore.

GetPixelColor(X, Y) Restituisce il colore del pixel specificato con X e Y.

Il colore viene restituito come numero intero a 24 bit che contiene il rosso negli otto bit più alti, il verde negli otto bit intermedi e il blu negli otto bit bassi. Se si converte questo numero con la funzione HEX si possono distinguere bene i tre colori.

- - - - - -

Le funzioni seguenti estraggono le caratteristiche dalla variabile numerica "colore" e restituiscono un valore numerico.

GetColorRed(colore) Restituisce la quantità di rosso (un numero da 0 a 255)

GetColorGreen(colore) Restituisce la quantità di verde (un numero da 0 a 255)

GetColorBlue(colore) Restituisce la quantità di blu (un numero da 0 a 255)

GetColorHue(colore) Restituisce la tinta (un numero da 0 a 360)

GetColorSaturation(colore) Restituisce la saturazione (un numero da 0 a 255)

GetColorLightness(colore) Restituisce la luminosità (un numero da 0 a 255)

- - - - - -

GetColorIndex(colore) Restituisce l'indice del colore (da 0 a 8), scelto tra i nove colori seguenti: Nero, Rosso, Arancio, Giallo, Verde, Ciano, Blu, Viola, Bianco.

- - - - - -

Le due funzioni seguenti estraggono il nome del colore dalla variabile numerica "colore" e restituiscono un valore di tipo stringa.

GetSimpleColorName(colore) Restituisce il nome del colore, scelto tra i nove colori seguenti: Nero, Rosso, Arancio, Giallo, Verde, Ciano, Blu, Viola, Bianco.

GetNearestColorName(colore) Restituisce il nome del colore conosciuto che più si avvicina.

- - - - - -

Vedere anche gli esempi nella cartella:
" Programs \ Demo Programs \ Demo GetColors "

La funzione Key()

La funzione Key() restituisce il valore booleano True se il tasto indicato è premuto e se la applicazione Automation è selezionata.

Se si desidera che l'istruzione Key agisca anche quando la applicazione Automation non è selezionata o, ad esempio, quando è minimizzata, si può utilizzare l'istruzione "Option GlobalKeys Enabled".

Vedere gli esempi "Wait_Key.txt" e "Option GlobalKeys.txt" che si trovano nella cartella "Examples\Demo Keys".

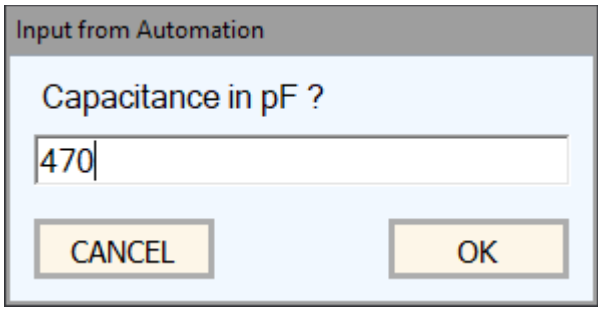
Notare che la istruzione Wait può attendere l'avverarsi di una condizione, per cui:

- Wait Key("Space") attende che la barra spazio venga premuta.
- Wait Not Key("Space") attende che la barra spazio venga rilasciata.

La funzione Input

Questa funzione mette in pausa il programma, e attende che l'utente immetta manualmente un valore.

Il valore può essere una stringa o un numero, a seconda che a sinistra dell'uguale si utilizzi una variabile stringa o numerica.



Se l'utente preme CANCEL la variabile non viene modificata.

Esempi:

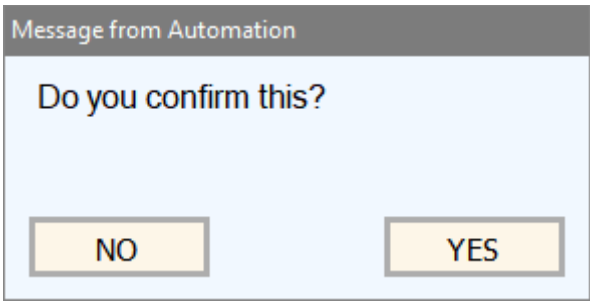
```
S1 = Input "Write your input here"
```

```
V1 = Input "Write a number"
```

Vedere anche gli esempi nella cartella
"Demo Programs \ Demo Input"

La funzione "Message"

Questa funzione presenta un messaggio e attende che l'utente faccia una scelta tra "YES" e "NO".



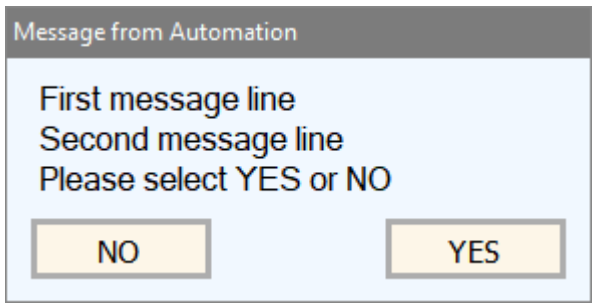
Esempi:

```
S1 = Message Do you confirm this?
V1 = Message Do you confirm this?
```

Se a sinistra dell'uguale si utilizza una variabile numerica allora il suo valore numerico sarà "0" per "NO" e "1" per "YES".

Se invece la variabile è una stringa conterrà direttamente "NO" o "YES".

Volendo si può scrivere il testo su più righe, utilizzando i caratteri CRLF, come nell'esempio seguente:




```
s1 =      "First message line" + CRLF
s1 = s1 + "Second message line" + CRLF
s1 = s1 + "Please select YES or NO"
v1 = Message s1
```

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo Input and Message”

Le funzioni di data e tempo

Date (Anno, Mese, Giorno) - Anno, mese e giorno convertiti in data

Now - Ritorna la data e l'ora dell'istante presente

Today - Ritorna la data dell'istante presente

Le funzioni che restituiscono una data (Date, Now e Today) possono essere usate nelle formule. Si possono sommare o sottrarre una dall'altra e si possono anche sommare o sottrarre con numeri. In questo caso i numeri rappresentano giorni e la parte decimale del numero rappresenta frazioni di giorni (che non sono minuti o secondi ma “frazioni di giorni” cioè, ad esempio, “0.5” giorni valgono 12 ore).

Bisogna fare attenzione che queste funzioni non danno delle stringhe ma numeri per cui in alcuni casi non funzioneranno, ad esempio **Print Left (Now , 2)** **non stampa le prime due cifre** dell'anno corrente ma stampa "**No**".

In questi casi bisognerà quindi trasformare la data in stringa ad esempio con: **Print Left (Str (Now), 4)** che stamperebbe **"2024"**.

La funzione "ElapsedTime"

ElapsedTime - Secondi dall'avvio del programma.

Questa funzione fornisce i secondi di tempo passati dall'avvio del programma. I secondi comprendono anche i decimali fino ai decimi di microsecondo.

Per misurare un intervallo di tempo (cronometro), si può segnare l'istante di inizio in una variable (da v1 a v99 o una variabile numerica definita con "Variable") e poi sottrarla dal tempo finale. Come nell'esempio seguente:

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

Se si tiene in funzione il programma a lungo, il valore ElapsedTime cresce molto, ma non ci si deve preoccupare di questo. Gli arrotondamenti numerici non causano perdita di precisione, e i microsecondi continuano ad essere validi, anche se si tiene in funzione il programma per molti anni.

Per memorizzare i tempi è bene usare v1..v99 oppure le variabili numeriche definite con "Numeric" o gli Array, che sono a doppia precisione (16 cifre significative), e non gli Slot che contengono numeri a singola precisione (8 cifre significative).

Le funzioni di filtro

Si usano queste funzioni per filtrare dati numerici rumorosi, esattamente come si farebbe con i classici filtri elettronici a resistenza e condensatore (HP = passa alto e LP = passa basso).

- **v2 = FilterHP (v1, Frequenza, NumeroPoli)**
- **v3 = FilterLP (v2, Frequenza, NumeroPoli)**

ATTENZIONE: queste funzioni devono essere eseguite
con frequenza almeno doppia della frequenza di taglio impostata
e se ne possono scrivere al massimo una per ogni riga di programma.

Impostando NumeroPoli = 1 si ottiene un filtro composto da un singolo resistore e un singolo condensatore, che fornisce una pendenza di 6 decibel per ottava (o in altre parole un dimezzamento della tensione per ogni raddoppio della frequenza (filtri passa basso) o per ogni dimezzamento della frequenza (filtri passa alto).

Aumentando NumeroPoli (fino a un massimo di 99) si ottengono filtri composti da molte celle in cascata, che hanno una pendenza e quindi un effetto di filtraggio notevolmente maggiori (fino a quasi 600 dB per ottava). Filtri di così alto ordine sarebbero impossibili da realizzare con componenti elettronici, non solo per le dimensioni esagerate che si otterrebbero, ma anche perché non funzionerebbero proprio, a causa del carico prodotto dalle celle successive su quelle precedenti.

Normalmente si utilizza un numero di poli non molto alto (da 1 a 10) ma se necessario si può aumentarlo e ottenere maggiore filtraggio. L'unico effetto collaterale dell'aumento dei poli è un aumento del tempo di transito dei segnali.

Esempi

`v2 = FilterHP(v1, 2.5, 1)` Questo è un filtro passa alto, con frequenza di taglio di 2.5 Hz e un singolo polo.

`v3 = FilterLP(v2, 1, 15)` Questo è un filtro passa basso, con frequenza di taglio di 1 Hz e quindici poli.

A causa dei quindici poli quest'ultimo filtro ha un tempo di transito di mezzo secondo, e con sessanta poli il tempo di transito salirebbe a un secondo. Ma il tempo di transito varia anche con l'inverso della frequenza. Per cui se la frequenza fosse di dieci hertz questi tempi scenderebbero a ventesimi e decimi di secondo.

Le funzioni dei "Media"

Dopo aver avviato suoni e video con l'istruzione "Load" può essere utile attendere la fine della esecuzione, oppure conoscere la lunghezza e la posizione attuale in secondi.

Le seguenti funzioni facilitano queste operazioni.

MediaLength
Risponde la lunghezza totale del file in esecuzione (in secondi)

MediaPosition
Risponde la posizione attuale del file in esecuzione (in secondi)

MediaPlaying
Risponde TRUE se il file è in esecuzione, oppure FALSE se l'esecuzione è finita.

Ecco un esempio che avvia un video e poi attende la fine della esecuzione:

```
Load Video1.avi
Wait Not MediaPlaying
```

- - - - - -

Gli esempi sono nella cartella "Demo LOAD and MEDIA functions".

Vedere anche i comandi `Load Hide` `Load Stop` `Load Pause` `Load Position` e `Load Play`, che si utilizzano per controllare l'esecuzione dei file, e che sono spiegati in ***questa pagina*** .

Le funzioni per i file XML

DecodeXML

La funzione **`XML = DecodeXML(XML, Sezione)`** accetta una stringa contenente un testo XML e restituisce una stringa che contiene solo la sezione di testo che inizia con **`<Sezione>`** e finisce con **`</Sezione>`**

Per isolare un valore si possono utilizzare più righe DecodeXML successive. La prima riga isolerà un blocco, la seconda un sotto-blocco e così via, fino ad ottenere il singolo valore.

Alcuni file XML contengono elenchi composti da più sezioni con lo stesso nome. In questi casi si utilizza un ciclo FOR che contiene una DecodeXML con un parametro aggiuntivo. Il parametro aggiuntivo, che è anche la variabile crescente del ciclo FOR, specifica quale sezione estrarre. I valori di questo parametro partono da uno e crescono fino a quando la DecodeXML risponde una stringa vuota, poi si utilizza un Exit per terminare il ciclo.

Vedere l'esempio "DecodeXML Plants"
e gli altri esempi della cartella "Demo Programs \ Demo XML"

FormatXML

La funzione **`XML = FormatXML(XML, Numero-spazi-iniziali)`** accetta una stringa contenente un testo XML e restituisce una stringa con gli spazi iniziali e i caratteri di cambio linea, che la rendono leggibile dagli umani.

Nell'esempio seguente sono stati usati cinque spazi iniziali. Se nella funzione **`FormatXML`** si omette il secondo parametro, allora vengono usati due spazi.

```
<planes><plane><year> 1977 </year><make> Cessna </make><model> Skyhawk </model><color> Light blue and white </color></plane><plane><year> 1975 </year><make> Piper </make><model> Apache </model><color> White </color></plane><plane><year> 1960 </year><make> Cessna </make><model> Centurian </model><color> Yellow and white </color></plane></planes>
```

```
<planes>
    <plane>
        <year> 1977 </year>
        <make> Cessna </make>
        <model> Skyhawk </model>
        <color> Light blue and white </color>
    </plane>
    <plane>
        <year> 1975 </year>
        <make> Piper </make>
        <model> Apache </model>
        <color> White </color>
    </plane>
    <plane>
        <year> 1960 </year>
        <make> Cessna </make>
        <model> Centurian </model>
        <color> Yellow and white </color>
    </plane>
</planes>
```

Le funzioni per i File e i Percorsi

SelectFile (FolderName)
Apre una finestra e restituisce una stringa con il percorso e il nome del file.

GetFileNames (FolderName)
Restituisce una stringa con percorso e nome di tutti i file separati da CRLF.

GetFileNames (FolderName, Sorted, Extension)
Come la precedente ma eventualmente in ordine alfabetico e selezionati.

GetFolderNames (FolderName)
Restituisce una stringa con tutti i percorsi, separati da CRLF.

GetFolderNames (FolderName, Sorted)
Come la precedente ma eventualmente in ordine alfabetico.

FileExists (FileName)
Restituisce TRUE se il file esiste.

AppRunning (ApplicationNameAndExtension)
Restituisce TRUE se la applicazione è in esecuzione,

GetTextFile (FileName)
Restituisce una stringa contenente il testo del file.

GetApplicationPath
Restituisce una stringa con il percorso dell'applicazione.
In alternativa si potrebbe usare ".\xxxx" per indicare il path di Automation.exe.

GetFilePath (FilePathAndName)
Restituisce una stringa con il percorso del file.

GetFileNameWithoutPath (FilePathAndName)
Restituisce una stringa con solo il nome del file.

NOTE
Tutte le stringhe che contengono percorsi non hanno la barra rovesciata finale.
I nomi di file e cartelle possono essere una variabile stringa oppure: "xxxx.yyy"
"Sorted" è una variabile booleana, con valori immediati si scrive: True o False.

- - - - - -

Vedere anche il file di esempio: "Demo Files\SelectAndGetFilesAndFolders.txt"

Auto indentazione

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

L'indentazione migliora la visibilità del programma, evidenzia l'inizio e la fine delle strutture, e rende più facile scrivere software senza errori.

La applicazione Automation indenta automaticamente i programmi, mentre li si scrivono.

Abituarsi a vedere il software indentato è un importante aiuto didattico.

Diventerà poi più facile indentare manualmente, anche utilizzando ambienti di programmazione che non dispongono di indentazione automatica.

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

Auto completamento dei costrutti

Quando si preme ENTER su una riga di tipo **For / If / Label o Select**, si ottiene il completamento automatico della struttura con le corrispondenti istruzioni **Next / EndIf / Return, Case e EndSelect.**

Prima di premere ENTER, la riga **For / If / Label o Select** deve essere completa e valida.

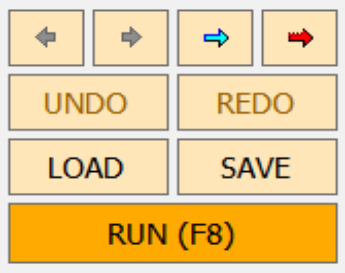
- - - - - -

Vedere anche gli esempi nella cartella
“Demo Programs \ Demo INDENT"

Menu e finestre

I pulsanti di controllo

Le funzioni principali sono sempre disponibili su questi pulsanti, che possono essere premuti con il Mouse, o con il dito su uno schermo tattile.



Il più importante è il RUN che avvia o ferma il programma.

Poi ci sono i pulsanti LOAD e SAVE, che servono per caricare e salvare i programmi (vedere anche ***questa pagina*** che spiega come modificare e salvare i programmi).

Il pulsante UNDO serve per tornare indietro, se si sono fatte modifiche al programma e si vuole eliminarle. Il pulsante REDO ricostruisce le modifiche eliminate con UNDO.

Le due FRECCE scure spostano il cursore, e anche la pagina visibile, sulle sezioni di programma visitate in precedenza.

La FRECCIA azzurra cerca tutte le occorrenze di funzioni, variabili o anche semplici parole.

La FRECCIA rossa cerca solo nelle dichiarazioni (Button, Key, Label e Variable)

Le funzioni di ricerca sono comode, basta selezionare una parola o anche solo posizionare il cursore su di essa e poi premere ripetutamente la freccia.

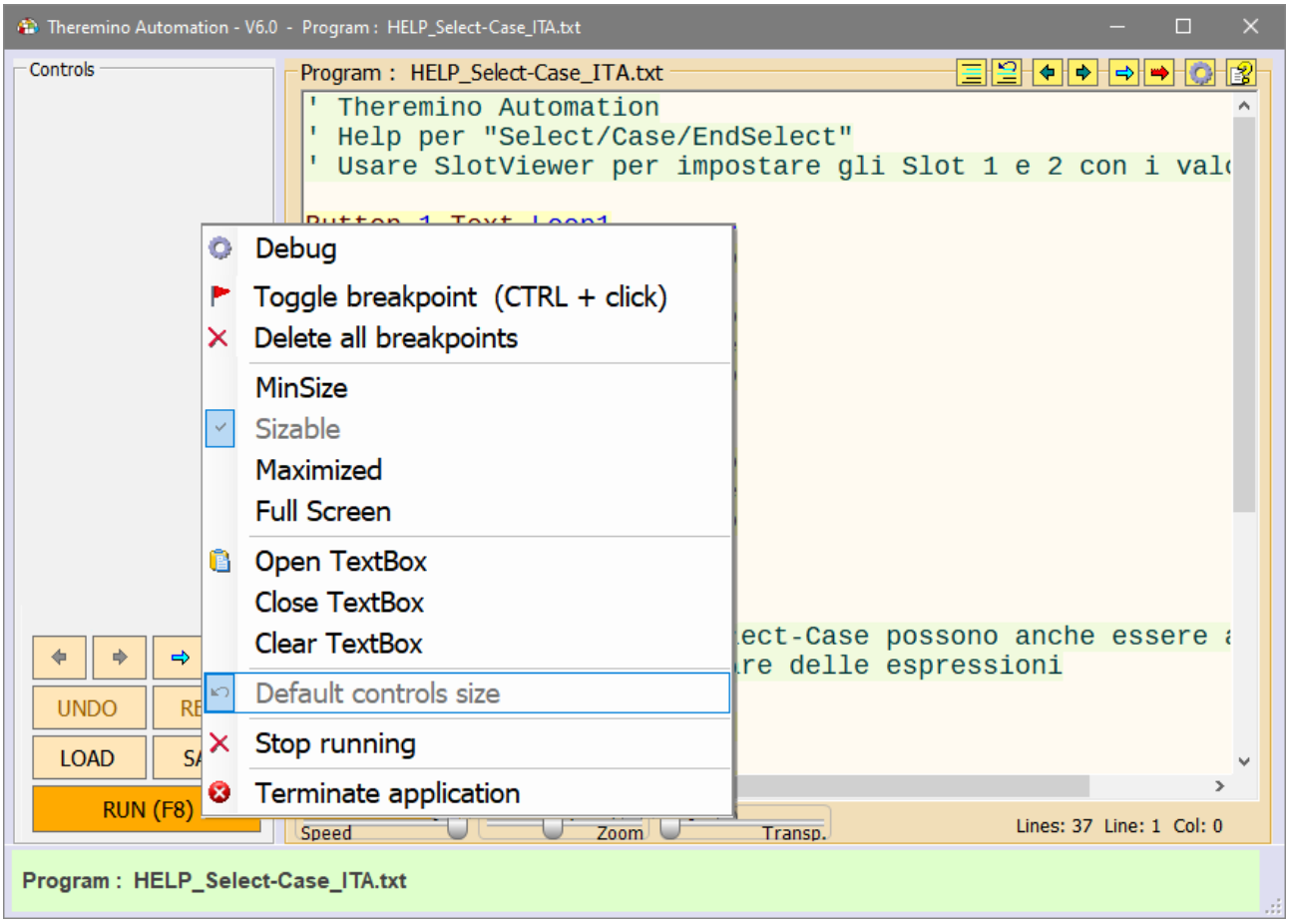
Nel caso si stia visualizzando un video o una immagine a tutto schermo, questi pulsanti non sono visibili.

In quelle occasioni, per interrompere il programma, si possono utilizzare i metodi seguenti:

- Si può interrompere l'esecuzione del programma in ogni momento, con il tasto **SHIFT-ESC,** oppure con **ALT-E.**
- Si può utilizzare il pulsante destro del mouse e aprire il Menu della applicazione (mostrato nella prossima pagina).
- Si può avviare e fermare l'esecuzione con **F8** o con **ALT-R**
- Quando la finestra di Debug è aperta **F8** e **ALT-R** eseguono un “RUN FROM CURSOR”, altrimenti un normale “RUN” dall'inizio del programma.

Il menu della applicazione

Facendo click con il tasto destro del Mouse (oppure toccando lo schermo tattile senza staccare il dito per due secondi), si apre il menu che si vede qui sotto.



Quando il programma è fermo, si deve fare click sulla parte sinistra della applicazione (zona dei controlli), e non sulla zona del programma.

Invece, quando il programma è avviato, questo menu appare cliccando dovunque, anche sul programma, nonché sulle immagini e i video a tutto schermo.

Questo menu permette di aprire la finestra di Debug, aggiungere e togliere Breakpoints, scegliere le dimensioni della finestra principale, Aprire, Chiudere e Svuotare la TextBox, fermare l'esecuzione del programma e anche chiudere totalmente la applicazione Automation.

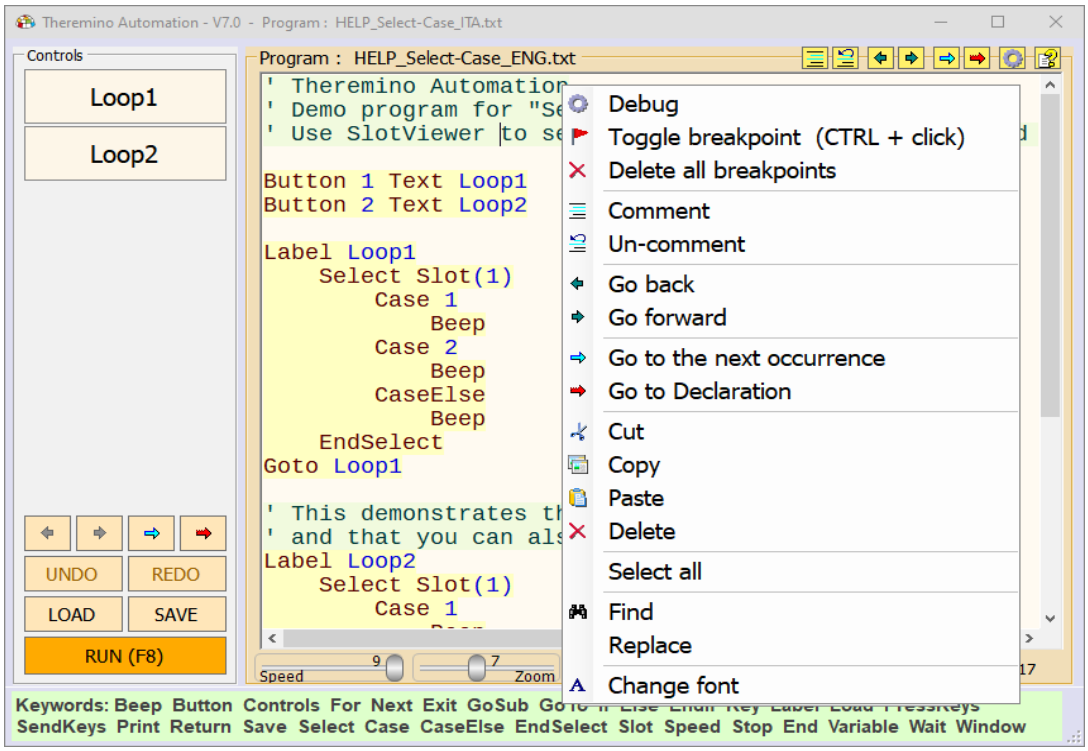
Conoscere questo menu è importante.

Senza di lui non si riuscirebbe più a uscire da alcune situazioni.

Ad esempio quando è attivo un video a tutto schermo.

Il menu del programma

Facendo click sulla zona del programma, con il tasto destro del Mouse (oppure toccando lo schermo tattile senza staccare il dito per due secondi), si apre il menu che si vede qui sotto.



Il programma non deve essere in funzione, altrimenti al posto di questo menu si aprirebbe quello della pagina precedente. Le prime tre righe servono per il Debug (manutenzione del software) e saranno spiegate meglio in ***questa pagina***.

"Comment" e **"Uncomment"** servono per commentare (aggiungere l'apice iniziale) a intere zone del programma. Oppure per eliminare i commenti.

"Go Back" e **"Go Forward"** spostano il cursore, e anche la pagina visibile, sulle sezioni di programma visitate in precedenza.

"Go to the next occurrence" cerca altre occorrenze della parola selezionata.

"Go to declaration" cerca la parola selezionata solo nelle righe di dichiarazione.

I comandi **"Cut"**, **"Copy"**, **"Paste"**, **"Delete"** e **"Select All"**, copiano, incollano, cancellano e selezionano parti del programma. Al loro posto si potrebbero anche usare i tasti CTRL-X, CTRL-C, CTRL-V, CANC e CTRL-A.

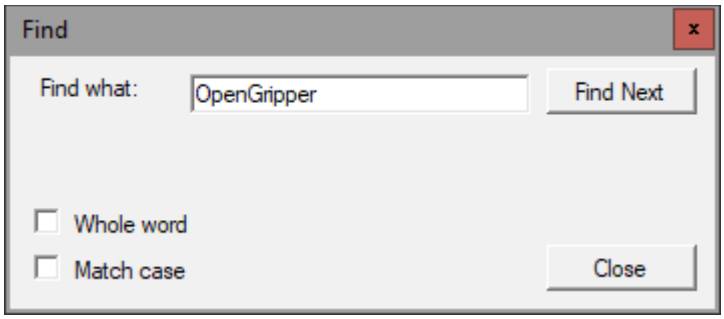
"Find" (o CTRL-F) e **"Replace",** aprono la finestra per cercare e sostituire parole e frasi.

L'ultima riga **"Font"** serve per scegliere il tipo di carattere. Facendo click su di essa si apre una lista di tipi di caratteri, sul lato sinistro della applicazione, e alcuni pulsanti di controllo.

Find e Replace

Le ultime voci in basso del menu della applicazione aprono due finestre simili tra loro.

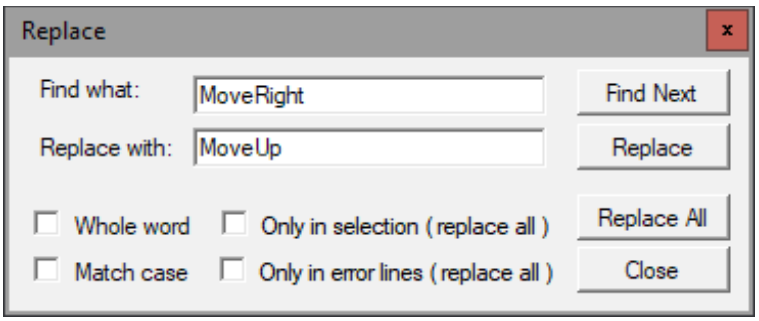
La finestra “FIND”



Con questa finestra si cercano parole o frasi nel testo del programma.

- Se si abilita “Whole word”, la parola deve essere completa.
- Se si abilita “Match case”, le parola deve corrispondere anche come maiuscole e minuscole.
- Con “Find next” (o con F3), si passa alla prossima occorrenza della parola cercata. Se si arriva alla fine del programma la ricerca riparte dall'inizio.

La finestra “REPLACE”



Questa finestra ha le stesse opzioni della precedente, ma permette anche di sostituire la parola (o la frase) con un altra.

Se si preme “Replace” si effettua una sola sostituzione. Invece con “Replace All” si sostituiscono tutte le occorrenze.

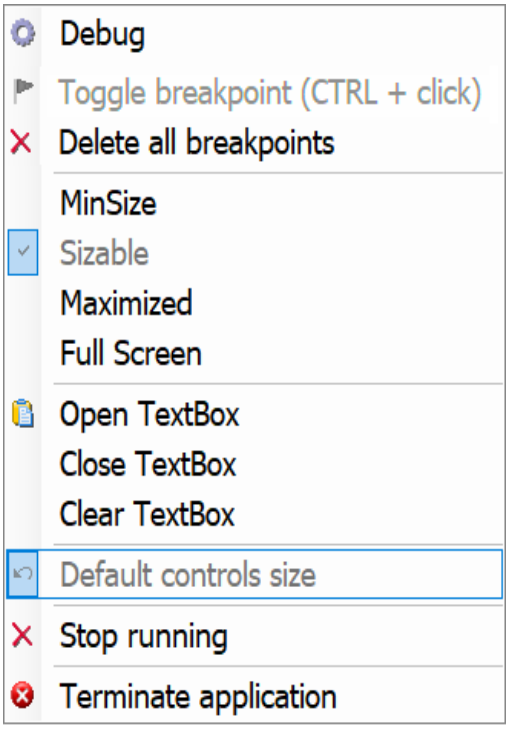
La sostituzione può avvenire in tutto il programma o soltanto nella zona selezionata, oppure soltanto nelle linee che contengono errori (o "warnings").

Funzionamento a tutto schermo

Con “tutto schermo” si intende che la finestra di visualizzazione non ha bordi visibili, e che le barre del desktop vengono nascoste.

Automation può visualizzare a tutto schermo le immagini e i video.

Anche le pagine web, e il testo del programma, vengono visualizzati a tutto schermo, ma con la barra verticale dei pulsanti sulla sinistra.



Per aprire la finestra a tutto schermo, si può utilizzare il menu che si apre con il pulsante destro del Mouse, e scegliere “**Full Screen**”.

Per tornare al funzionamento in finestra, si scelgono le opzioni “**Maximized**”, “**Sizable**” o "**MinSize**".

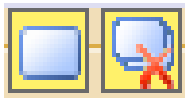
In alternativa, per uscire dalla condizione di FullScreen, si potrebbe interrompere l'esecuzione del programma, con i tasti **SHIFT-ESC** oppure **ALT-E.**

Durante l'esecuzione del programma si possono anche utilizzare le istruzioni `Window MinSize`, `Window FullScreen`, `Window Maximized` e `Window Sizable`, spiegate in ***questa pagina***.

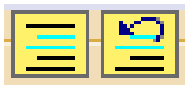
- - - - - -

Per sperimentare con la istruzione “Windows”,
e con le varie dimensioni della finestra,
vedere l'esempio “Demo Programs \ Demo – Windows”.

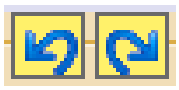
I controlli della barra superiore



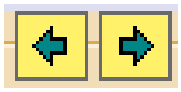
Il primo pulsante aggiunge o toglie un segnalibro. Con il secondo li si possono eliminare tutti.



Questi due commentano e de-commentano il testo selezionato.



Le due frecce blu servono per tornare indietro nelle modifiche al programma e per ricostruire le modifiche eliminate.

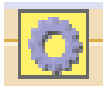


Le due FRECCE scure spostano il cursore, e anche la pagina visibile, sulle sezioni di programma visitate in precedenza.

La FRECCIA azzurra cerca tutte le occorrenze di funzioni, variabili o anche semplici parole.

La FRECCIA rossa cerca solo nelle dichiarazioni (Button, Key, Label e Variable)

Le funzioni di ricerca sono comode, basta selezionare una parola o anche solo posizionare il cursore su di essa e poi premere ripetutamente la freccia.



L'ingranaggio apre la finestra di Debug.

Il punto interrogativo apre il file di istruzioni (Help) nella lingua prescelta. Per far funzionare questo comando si deve copiare il file di Help della lingua preferita nella cartella "Automation\Docs". I file di Help più recenti si scaricano da ***questa pagina***.

Se il file di Help non viene trovato allora appare un messaggio che suggerisce di aprire la cartella Docs e copiarvi il file.

Oppure si può scegliere di selezionare un file di Help nella lingua preferita posizionato nella cartella "Docs" o in qualunque altra cartella. *Per cambiare il file selezionato cliccare il pulsante con il tasto destro del mouse*.

I controlli della barra inferiore

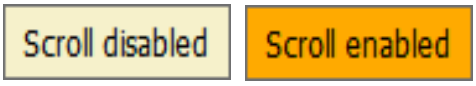
Questo cursore regola la velocità di esecuzione.

Le velocità vanno da ”1” (una istruzione al secondo), fino a ”8” (diecimila istruzioni al secondo), e a ”9” (la massima velocità permessa dal sistema).

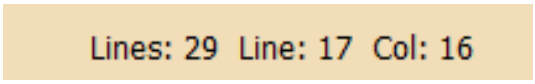
Mentre si scrive il programma è bene utilizzare una velocità media. Di solito la velocità ”5” (20 istruzioni al secondo), che è abbastanza lenta da poter seguire visivamente l'esecuzione del programma.

Il cursore ZOOM stabilisce la dimensione del testo, sia nella finestra del programma, sia in quella di Debug.

Questo cursore regola la trasparenza della finestra principale e permette di vedere anche sotto di essa.



Questo pulsante abilita lo "scroll" automatico del programma durante l'esecuzione. Lo si tiene disabilitato per potersi concentrare su una funzione. Lo si abilita invece quando si vuole seguire l'andamento generale del programma.

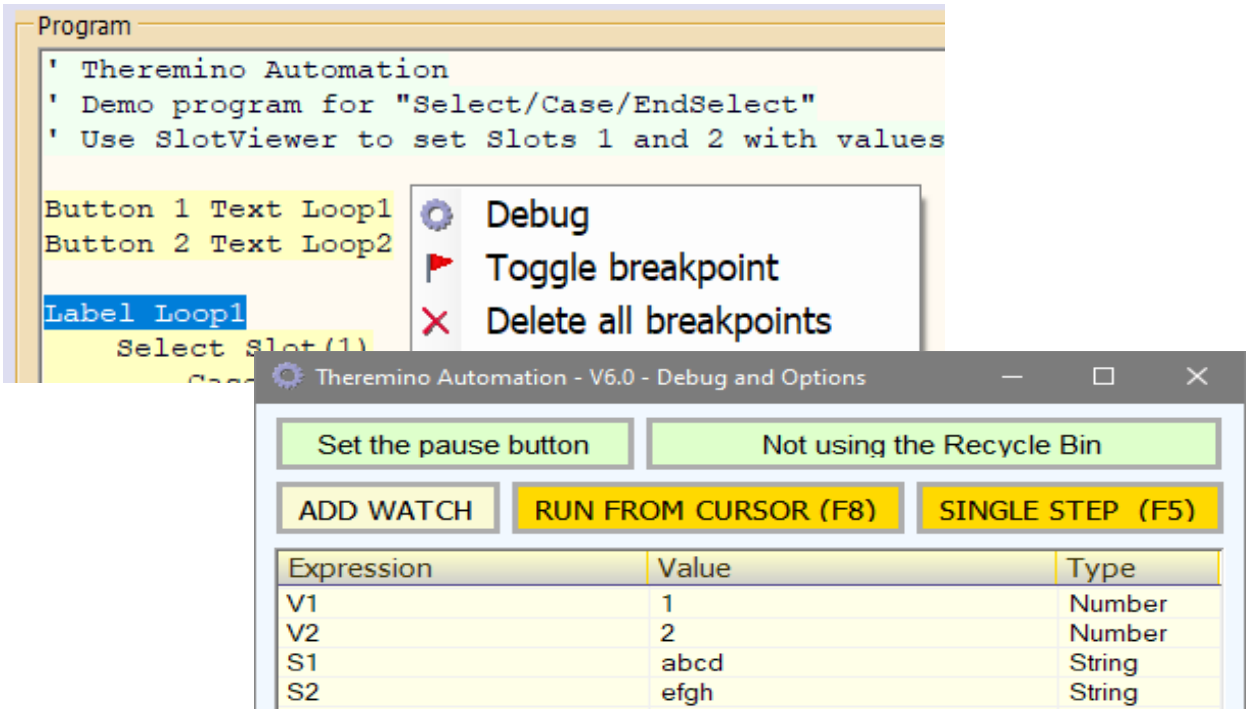


La parte destra della barra inferiore mostra informazioni sul programma:

- Il numero totale di linee
- La linea dove si trova il cursore (partendo da linea 1)
- La colonna dove si trova il cursore (partendo da colonna 1)

La finestra di Debug

La finestra di “Debug” facilita la messa a punto del software. Per aprirla si preme il pulsante destro del Mouse, sulla zona del programma, e si sceglie “Debug”.



Le due funzioni principali sono: “RUN FROM CURSOR”, che permette di eseguire il programma da qualunque punto, e “SINGLE STEP”, che permette di eseguirlo una riga per volta.

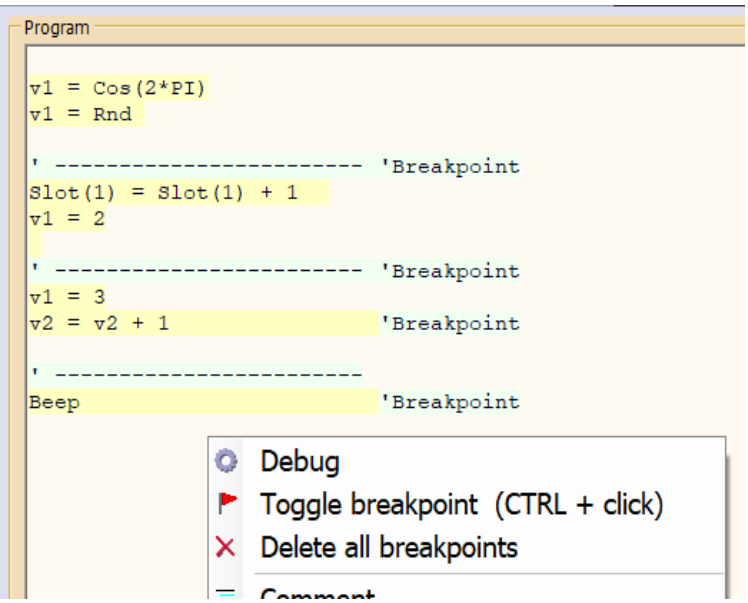
Per eseguire il programma da un punto a piacere, prima di tutto lo si ferma, con il tasto STOP della finestra principale. Oppure si può fermarlo, premendo RUN FROM CURSOR (**F8**) o SINGLE STEP (**F5**). Poi si posiziona il cursore sulla riga che si vuole eseguire, e si preme uno di questi due pulsanti.

Quando la finestra di Debug è aperta il tasto **F8** esegue sempre il “Run from cursor”, mentre quando è chiusa esegue il RUN da inizio programma.

Le altre funzioni di questa finestra sono: i Watch (espressioni di controllo), la riga di exec (eseguire istruzioni), i BreakPoint (punti di interruzione), e due tasti di opzioni (di colore verde) che saranno spiegati nelle prossime pagine.

Sperimentate con gli esempi della cartella “Demo Debug”

La finestra di Debug (Breakpoints)



Il **tasto CONTROL + click sinistro del mouse** aggiunge o elimina i breakpoint.

Si può utilizzare questo metodo anche mentre il programma è in esecuzione.

Inoltre quando si aggiunge un breakpoint la finestra di Debug si apre automaticamente e questo è molto comodo per intercettare il programma durante l'esecuzione.

Quando il programma non è in esecuzione si possono modificare i breakpoint con il menu che si apre cliccando sul programma col tasto destro del mouse.

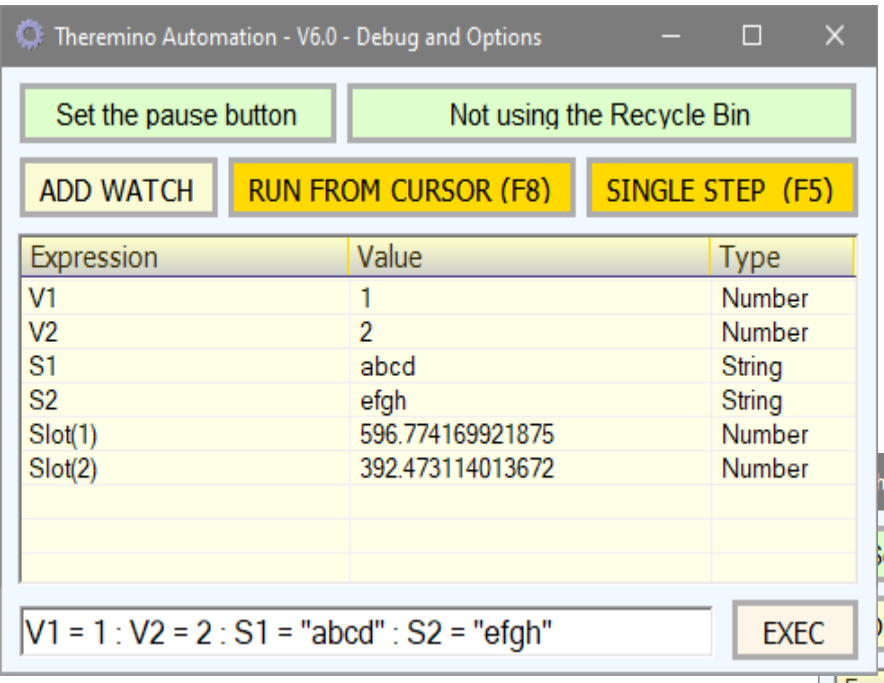
Quando la finestra di Debug è aperta, il programma si ferma ad ogni linea che termina con `'Breakpoint` .

Invece, se la finestra di Debug è chiusa, i Breakpoint vengono ignorati. Per cui è possibile lasciarli nelle posizioni preferite, anche nel programma definitivo.

Quando il programma è fermo ad un Breakpoint, si possono utilizzare tutte le opzioni della finestra di Debug.

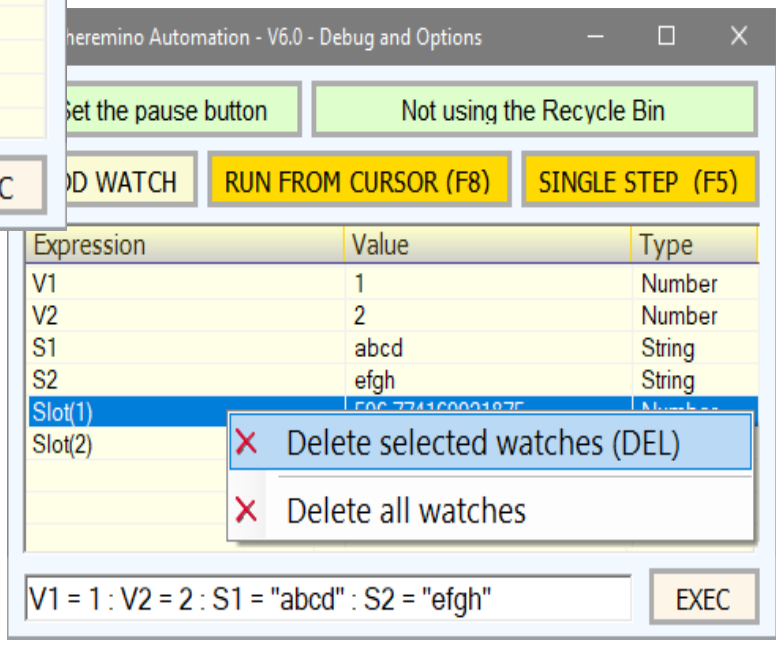
- Si possono esplorare i valori delle variabili, e degli Slot (da 0 a 999), con la tabella dei Watch.
- Nei Watch si possono anche scrivere espressioni complesse e calcoli.
- Si possono modificare i valori delle variabili e degli Slot, scrivendo assegnamenti nella riga di EXEC
- Si possono eseguire istruzioni, scrivendole nella riga e premendo EXEC
- Si può far continuare l'esecuzione, con RUN FROM CURSOR
- Si può eseguire una singola linea, con SINGLE STEP
- Si può modificare la posizione di esecuzione, e poi continuare con RUN FROM CURSOR o con SINGLE STEP

La finestra di Debug (Watches)



Ogni riga di questa tabella è un "Watch" (espressione di controllo).

Con i watch si osservano i valori delle variabili e il loro tipo, e si controlla il funzionamento delle espressioni del programma.



I Watch si aggiungono selezionando una variabile, funzione, o espressione del programma, e poi premendo il pulsante "ADD WATCH".

Per eliminare uno o più Watch, si selezionano le righe prescelte, si preme il tasto destro del mouse sulla tabella, e si utilizza il menu visibile qui a destra.

Per modificare un Watch si fa doppio click, con il tasto sinistro del Mouse, sulla sua riga, e si modifica il testo della espressione con la tastiera.

I Watch della applicazione Automation sono molto potenti, si possono scrivere espressioni complesse, fare dei calcoli, controllare se le espressioni sono vere o false, leggere i valori degli Slot, sommare stringhe di testo, ecc...

Inoltre, a differenza di quasi tutti gli ambienti di programmazione, i Watch vengono aggiornati in tempo reale (dieci volte al secondo), anche col programma fermo. Per cui rispecchiano sempre il valore attuale delle espressioni e degli Slot. Nemmeno DotNet fa questo, ed è una caratteristica molto utile.

- - - - -

Sperimentate con gli esempi della cartella "Demo Debug"

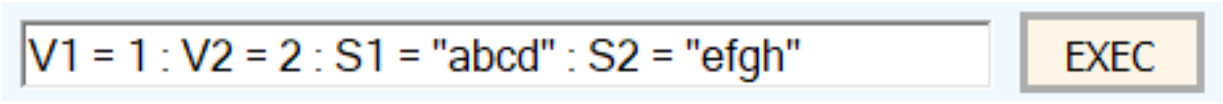
La finestra di Debug (Exec)

Con la riga inferiore della finestra di Debug, si possono eseguire istruzioni e assegnare valori alle variabili e agli Slot.

Utilizzando la parola Print, si possono stampare risultati di espressioni complesse, e quindi farsele calcolare e conoscere il risultato.

Si possono anche scrivere dei Goto e Gosub, e quindi premere EXEC e saltare alle etichette corrispondenti, anche mentre il programma è in funzione.

La riga accetta anche più istruzioni sulla stessa riga (separandole con i due punti, come si vede nell'esempio qui sotto).



Si possono utilizzare praticamente tutte le istruzioni che potrebbero essere scritte nel programma.

Le uniche parola chiave non valide sono: If, Else, Endif, Select, Case, CaseElse, EndSelect, Return, Stop e Wait. Scrivendole non si ottiene nessun effetto, semplicemente non vengono eseguite.

- - - - -

Quando si scrive nella riga di EXEC, gli errori e i suggerimenti vengono mostrati nella riga inferiore della finestra principale della applicazione.

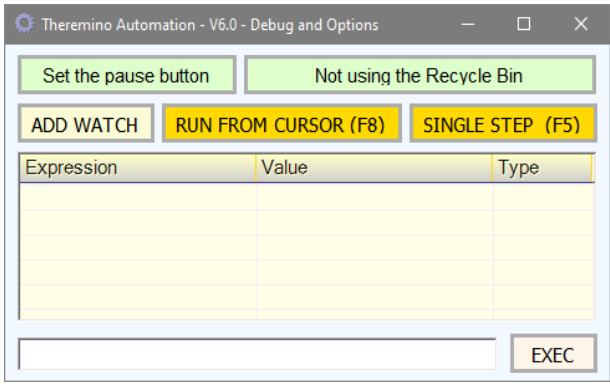
Cliccando sul pulsante "Select next errore" si evidenziano tutte le righe con errori, una dopo l'altra. Maggiori informazioni sugli errori e i suggerimenti nella pagina "***Parole chiave***".

- - - - -

Sperimentate con gli esempi della cartella "Demo Debug"

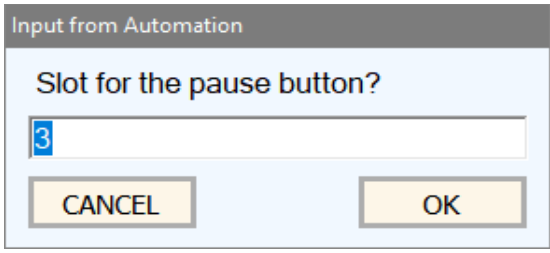
La finestra di Debug (Pause Button)

La opzione "PAUSE BUTTON SLOT" abilita un pulsante meccanico esterno per mettere in pausa l'esecuzione del programma.



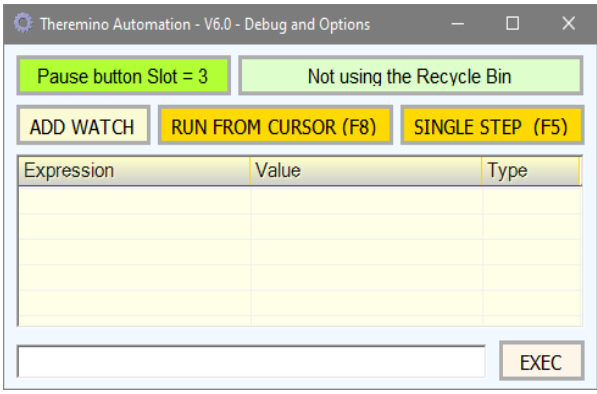
Per abilitare questa opzione si apre la finestra di debug, come spiegato nelle pagine precedenti.

Poi si fa click sul pulsante "Set the pause button" e si scrive un numero.



Questo numero (da 1 a 999) deve corrispondere allo Slot a cui si collega un pulsante esterno.

Per abilitare il pulsante esterno si scrive un numero da 1 a 999 e poi si preme OK. Per disabilitarlo si scrive uno zero, o si lascia la casella bianca.



Quando il pulsante esterno è collegato a uno Slot il pulsante "Pause Button Slot" diventa di colore verde scuro e indica il numero dello Slot impostato.

Premendo il pulsante esterno l'esecuzione del programma si ferma. Rilasciandolo riprende.

Normalmente la pausa si attiva quando il valore dello Slot è maggiore di 500, ma è anche possibile definire un valore diverso, ed anche se la Pausa deve essere attivata per valori maggiori o minori di questo. Gli esempi seguenti lo spiegano meglio:

12	La pausa si attiva quando il valore dello Slot 12 è maggiore di 500
12>300	La pausa si attiva quando il valore dello Slot 12 è maggiore di 300
12<10	La pausa si attiva quando il valore dello Slot 12 è inferiore a 10

Durante la pausa tutti i bordi della applicazione si colorano di arancione e si viene anche avvertiti con un messaggio nella riga inferiore della applicazione.

```
WARNING : Pause button is pressed (verify the Slot 3)
```

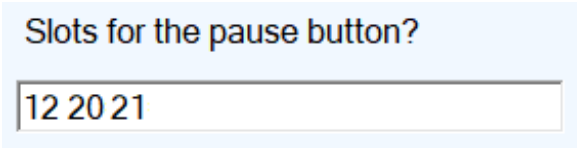
La finestra di Debug (Pause Button con LED)

Quando si utilizzano grosse macchine a volte si fanno lavorazioni utilizzando comandi vicini alla macchina (joysticks e pulsanti) e non sempre si guarda il monitor. In alcuni casi il monitor potrebbe anche essere spento e tutta la interazione avvenire guardando come si muove la macchina e per mezzo di suoni e di luci (solitamente LED tricolore o semafori) che indicano i vari stati della macchina (OK, attenzione, errori, ecc).

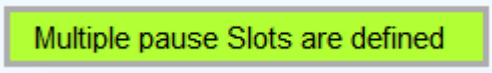
Può quindi accadere di avere il pulsante di Pausa premuto e non saperlo, e di conseguenza immaginare difetti che non esistono, Ci hanno quindi chiesto di poter colorare i LED con un colore a piacere e anche di farli lampeggiare quando il pulsante di Pausa è premuto.

Innanzi tutto per poterli regolare in luminosità i LED dovranno essere collegati a Slot configurati come Pwm16. E si consiglia anche di impostarli come logaritmici per avere una regolazione più lineare, simile a quella che vede l'occhio umano.

Poi si dovrà premere il pulsante "Set pause button" e impostarlo con più di un numero. Il primo numero (12 in questo esempio) indica lo Slot del pulsante Pausa, mentre i numeri che seguono indicano gli Slot dei LED.



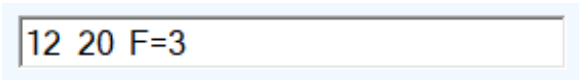
Se si impostano gli Slot dei LED il pulsante verde diventa come in questa immagine.



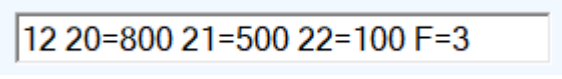
L'esempio "12 20 21" indica di accendere a massima luce i LED collegati agli Slot 20 e 21, quando si preme il pulsante, e di spegnerli quando lo si rilascia.

Il numero di LED da accendere dipende da quanti numeri si scrivono, se ne possono accendere uno, due, tre o anche di più.

Si possono anche far lampeggiare i LED con una frequenza da 0.2 a 20 Hertz, scrivendo alla fine "F=" seguito da un numero. L'esempio seguente fa lampeggiare il LED collegato allo Slot 20 con una frequenza di 3 Hertz.

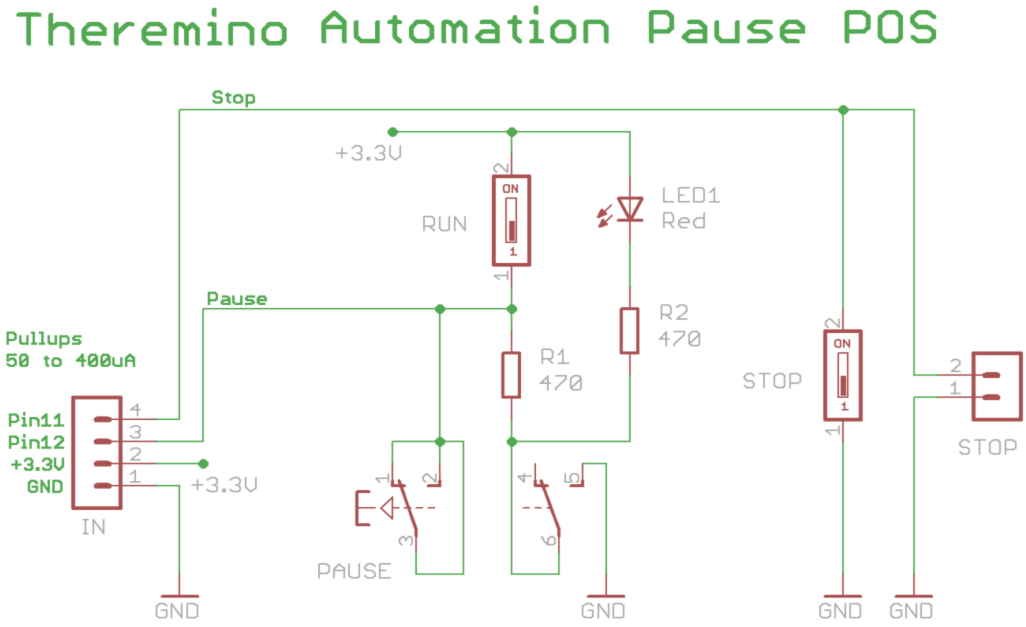


E infine si può anche stabilire la luminosità di ogni LED. L'esempio seguente accende il LED collegato allo Slot 20 con luminosità 800, il 21 con luminosità 500 e il 22 con luminosità 100 e li fa anche lampeggiare. Se si utilizza un LED tricolore il risultato è una luce Arancione che lampeggia velocemente.



La finestra di Debug (Pause Button adapters)

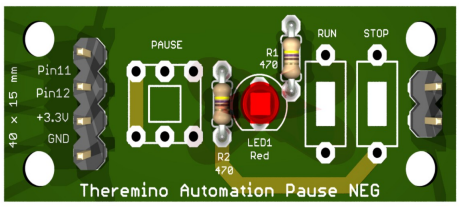
Per collegare i pulsanti di Pausa e di Stop abbiamo preparato dei piccoli adattatori in tre versioni. La prima versione (positiva) è adatta a pulsanti di pausa normalmente aperti, la seconda (negativa) per quelli normalmente chiusi e la terza è solo una serie di fori, utile per chi volesse costruire adattatori speciali.

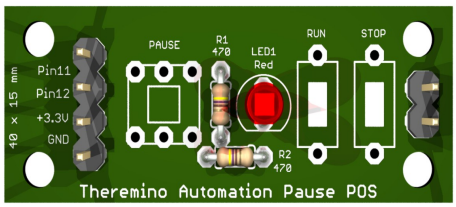


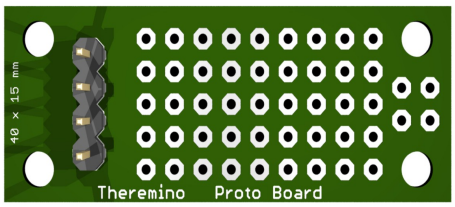
Il pulsante PAUSE è del tipo con ritenuta, cioè rimane premuto da solo.

Il pulsante RUN serve per sbloccare temporaneamente la pausa.

Il pulsante STOP serve per terminare il programma, ma attenzione che non viene gestito direttamente dalla applicazione Automation. Quindi si dovranno scrivere delle righe di programma per leggerlo e gestirlo.







I progetti di questi adattatori con schemi elettrici, immagini 3D e file di Eagle per fare i circuiti stampati si scaricano con questo collegamento: ***PauseAdapters***

Se li volete acquistare già montati, o eventualmente solo i PCB o i KIT di componenti, cercateli su ***www.store-ino.com*** oppure su eBay dal venditore ***maxtheremino***.

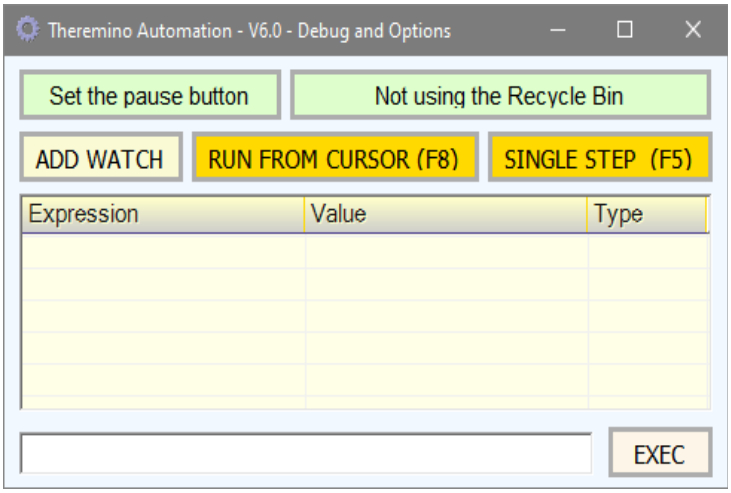
La finestra di Debug (Recycle Bin)

Ogni volta che si modifica il testo del programma e lo si esegue, oppure si carica un altro programma o si chiude la applicazione Automation, il programma corrente viene salvato su disco.

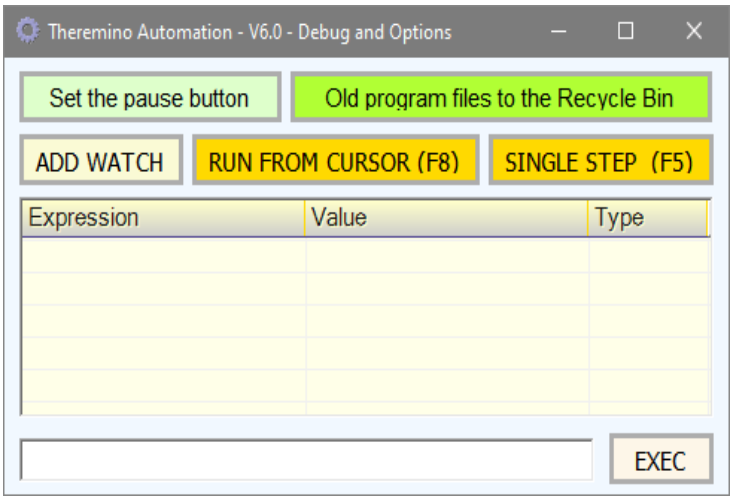
Potrebbe accadere di aver fatto modifiche involontarie o di aver fatto degli errori e di accorgersene solo il giorno seguente. Oppure potrebbe succedere di cancellare un programma per sbaglio.

In questi casi si vorrebbe recuperare il programma precedente che è stato però sovrascritto dalla nuova versione o cancellato.

La nuova opzione "Recycle Bin", introdotta dalla versione 6 di Automation, salva nel cestino ogni versione precedente prima di sovrascriverla.



Per abilitare l'uso del cestino si fa click sul pulsante "Not using recycle bin".



Quando questa opzione è abilitata il pulsante diventa verde più scuro e il testo cambia in "Old programs to recycle bin"

Sarà quindi possibile recuperare tutte le versioni precedenti, aprendo il cestino del sistema operativo, trascinando i file (uno per volta) in una cartella e aprendoli uno per uno con NotePad, fino a individuare la versione desiderata.

Tecniche di programmazione

Alcune istruzioni del linguaggio Automation eseguono operazioni che con altri linguaggi richiederebbero intere pagine di programma. Per esplorare tutte le possibilità del linguaggio consigliamo di caricare, leggere ed eseguire gli esempi che si trovano nelle cartelle:

Simple programs
Programmi di poche righe, tanto per iniziare a vedere qualcosa che si muove.

Demo Programs
Questa cartella contiene gli esempi più importanti, che spiegano tutte le parole chiave del linguaggio.

Advanced Programs
Qui si trovano i programmi più complessi che sono stati creati. Programmi anche da 1000 o 2000 righe, al limite delle possibilità di Automation.

Gli esempi più completi e recenti sono "**WXM_CNC_Vslot_Automation**" e "**WXM_CNC_Vslot_longrail**" che sono nella cartella "Advanced Programs" (per far funzionare la sequenza, e i LOG, dovrete anche utilizzare il tasto azzurro "Select sequence" e scegliere i file "CNC_PP_Cycle" o "LinearTest").

Questi programmi attendono il segnale di azzeramento della macchina all'avvio e tutte le volte che si fa partire la sequenza. Se non avete l'hardware potete superare questi blocchi **premendo a lungo il tasto CTRL**. E potete anche **commentare le righe** che fanno i **test: Master, RS485 e TestRAW**

Molti di questi programmi non funzionano o funzionano solo parzialmente perché avrebbero bisogno di un hardware specifico e anche di una struttura di cartelle e di file fatti apposta per loro. Ma possono essere utili per studiarne le tecniche e per copiare alcune funzioni da usare nei propri programmi.

- - -

Le prossime pagine spiegano alcune delle tecniche messe a punto durante la creazione di complessi programmi di automazione. Queste tecniche possono essere utili in alcuni casi, ma per utilizzarle bisogna avere un buon livello di esperienza nella programmazione.

Applicazioni e cartelle speciali

Nella stessa cartella dove si trova il file principale di questa applicazione, che si chiama "Theremino_Automation.exe", troverete anche una cartella con nome "Apps", che contiene altre applicazioni del sistema Theremino.

Queste applicazioni sono utilizzate da molti esempi di Automation. I programmi spesso le caricano all'avvio e le lasciano chiudere automaticamente alla chiusura di Automation.

Alcune delle applicazioni più spesso utilizzate sono:

Theremino_HAL per comunicare con i Master via USB.

Theremino_SlotViewer per visualizzare gli Slot e modificarli nelle prove.

Theremino_SignalScope per visualizzare gli Slot come con un oscilloscopio.

Tutte queste applicazioni devono stare nella cartella "Apps", per poter condividere il file "SlotNames.txt" che contiene i nomi e le opzioni degli Slot.

A volte si utilizzano più copie di una applicazione, soprattutto dello SlotViewer e del SignalScope e in tal caso si aggiunge un numero alla fine del nome, come in questo esempio: "Theremino_SlotViewer1.exe".

Importante
Le applicazioni che si trovano nella cartella APPS
potrebbero non essere le versioni più recenti.
Vanno bene per provare gli esempi di Automation,
ma per utilizzarle in nuovi programmi è consigliabile sostituirle
con le ultime versioni che si scaricano dal sito Theremino.

Aprire files con Notepad e altre applicazioni

Questi esempi aprono alcune applicazioni del sistema Windows

```
Label OpenNotepad
     Load "C:\Windows\Notepad.exe"
Return

Label OpenCalculator
     Load "C:\Windows\system32\Calc.exe"
Return

Label OpenPaint
     Load "C:\Windows\system32\mspaint.exe"
Return
```

Sperimentate con l'esempio "Open Notepad and Apps"
che si trova nella cartella "Demo Programs \ Demo Files"

Aprire applicazioni, ad esempio Notepad, indicando anche un file da aprire è più complesso, si consiglia quindi di preparare delle variabili che contengono i percorsi e i nomi dei file e poi di usarle con Load NomeVariabile, come nell'esempio seguente:

```
Label InitNotepadPaths
       Variable String EditorFolder
       Variable String NotepadApp
       Variable String EditSequence1
       EditorFolder = "Apps\Theremino_Editor\"
       NotepadApp = "C:\Windows\Notepad.exe"
       EditSequence1 = NotepadApp + " " + EditorFolder + "Sequences\CircleTest.seq"
Return

Label OpenNotepad
       Load NotepadApp
Return

Label OpenSequence1
       Load EditSequence1
Return
```

Sperimentate con l'esempio "Open Notepad With Files"
che si trova nella cartella "Demo Programs \ Demo Files"

Aprire cartelle

Questi esempi aprono alcune cartelle della applicazione Automation. Notare che il punto + barra rovesciata indica la cartella principale della applicazione Automation, cioè la cartella dove risiede il file "Automation.exe"

```
Label OpenMainFolder
        Load ".\"
Return

Label OpenFilesFolder
        Load ".\Files"
Return

Label OpenMediaFolder
        Load ".\Media"
Return
```

Per indicare cartelle e sottocartelle della applicazione Automation il punto e la barra si possono omettere, per cui i tre esempi precedenti possono diventare semplicemente:

```
Load ""    Load "Files"    Load "Media"
```

Per indicare la cartella superiore si utilizza il doppio punto. Ad esempio per aprire la cartella che contiene la cartella del file Automation.exe si può scrivere:

```
Load ".."
```

I prossimi esempi aprono il disco C e alcune cartelle del sistema Windows.

```
Load "C:\"

Load "C:\windows"

Load "C:\windows\system32"
```

- - - - -

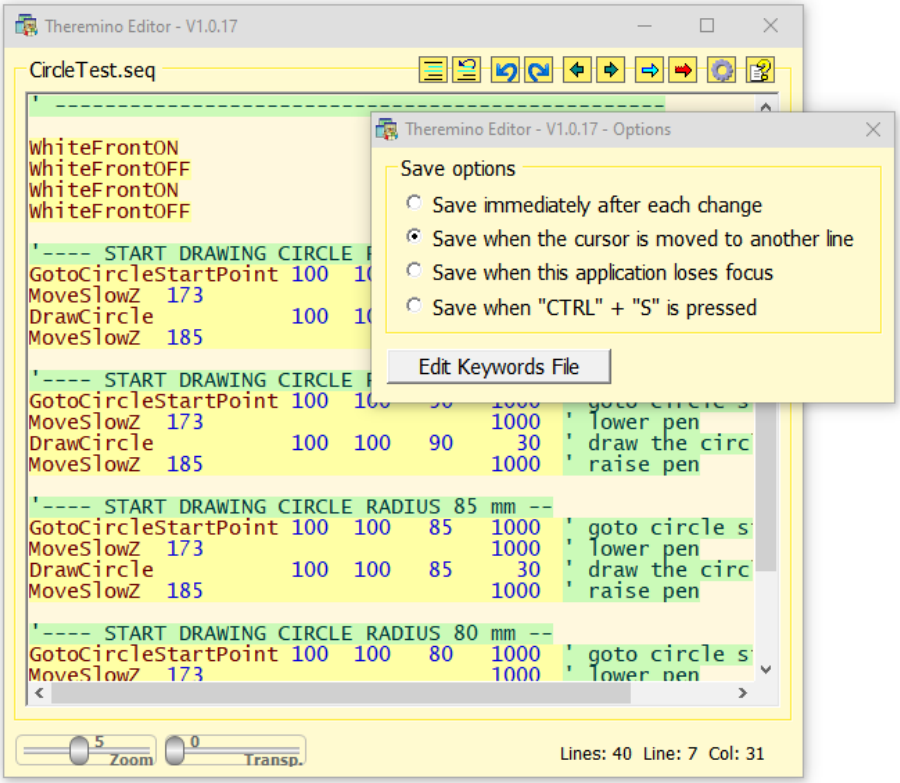
Sperimentate con l'esempio "Open Folders"
che si trova nella cartella "Demo Programs \ Demo Files"

L'applicazione "Theremino Editor"

Con questa applicazione si possono modificare sequenze di comandi durante l'esecuzione di un programma di Automation. Vedere nelle prossime pagine come si usano le sequenze.

La particolarità di questo editor è di modificare il file di sequenza mentre si scrive, senza quindi la necessità di utilizzare ogni volta il comando "Salva". Per cui si possono fare modifiche "in corsa", senza dover fermare e far ripartire l'esecuzione del programma di Automation.

Facendo click sul nome della sequenza, nella prima riga in alto si apre la cartella con i file di sequenza e li si possono copiare, rinominare e cancellare.



Nelle opzioni che si aprono facendo click sull'ingranaggio si può anche scegliere di salvare solo quando si passa a una riga differente, o solo quando si clicca su un'altra applicazione, oppure quando si preme CTRL + S.
Sempre nelle opzioni che si aprono con l'ingranaggio c'è il tasto "Edit Keywords File" che apre Notepad con un elenco di parole chiave.

Si possono aggiungere nuove parole all'elenco per mostrare gli errori con un colore diverso e quindi facilitare la scrittura delle sequenze.

Alcuni comandi possono essere composti da una sola parola, altri prevedono anche dei parametri. Troverete molti comandi di esempio, se non servono eliminateli e scrivete i vostri.

ATTENZIONE: I comandi che si scrivono nelle sequenze non vengono eseguiti automaticamente, ma dovranno essere interpretati nei "CASE" dell'esecutore di sequenze che si scrive nel programma di Automation.

Sperimentate con l'esempio "Demo OPEN EDITOR"
che si trova nella cartella "Demo Programs \ Demo Sequences"

Lanciare "Theremino Editor" con files

Lanciare la applicazione Theremino_Editor passando il nome della sequenza come parametro della riga di comando richiede una riga che contiene due percorsi separati da uno spazio. Difficilmente si riesce a scrivere tutto in una riga sola senza fare errori, per cui si consiglia di preparare delle variabili con i percorsi, come in questo esempio:

```
Label InitEditorPaths
        Variable String EditorFolder
        Variable String EditorApp
        EditorFolder = "Apps\Theremino_Editor\"
        EditorApp = EditorFolder + "Theremino_Editor.exe"
Return

Label OpenSequence1
        Load EditorApp + " " + EditorFolder + "Sequences\CircleTest.seq"
Return

Label OpenSequence2
        Load EditorApp + " " + EditorFolder + "Sequences\TestSeq.seq"
Return
```

Sperimentate queste tecniche con l'esempio "Demo OPEN EDITOR"
che si trova nella cartella "Examples\Demo Sequences"

Aprire le sequenze con Notepad

Si potrebbero anche aprire i file delle sequenze con Notepad. Utilizzare Notepad è facile ma rispetto a Theremino_Editor si perde la segnalazione degli errori e inoltre dopo ogni modifica si deve ricordarsi di salvare il file.

Ecco due esempi che aprono le sequenze con Notepad:

```
Load "Apps\Theremino_Editor\Sequences\CircleTest.seq"

Load "Apps\Theremino_Editor\Sequences\TestSeq.seq"
```

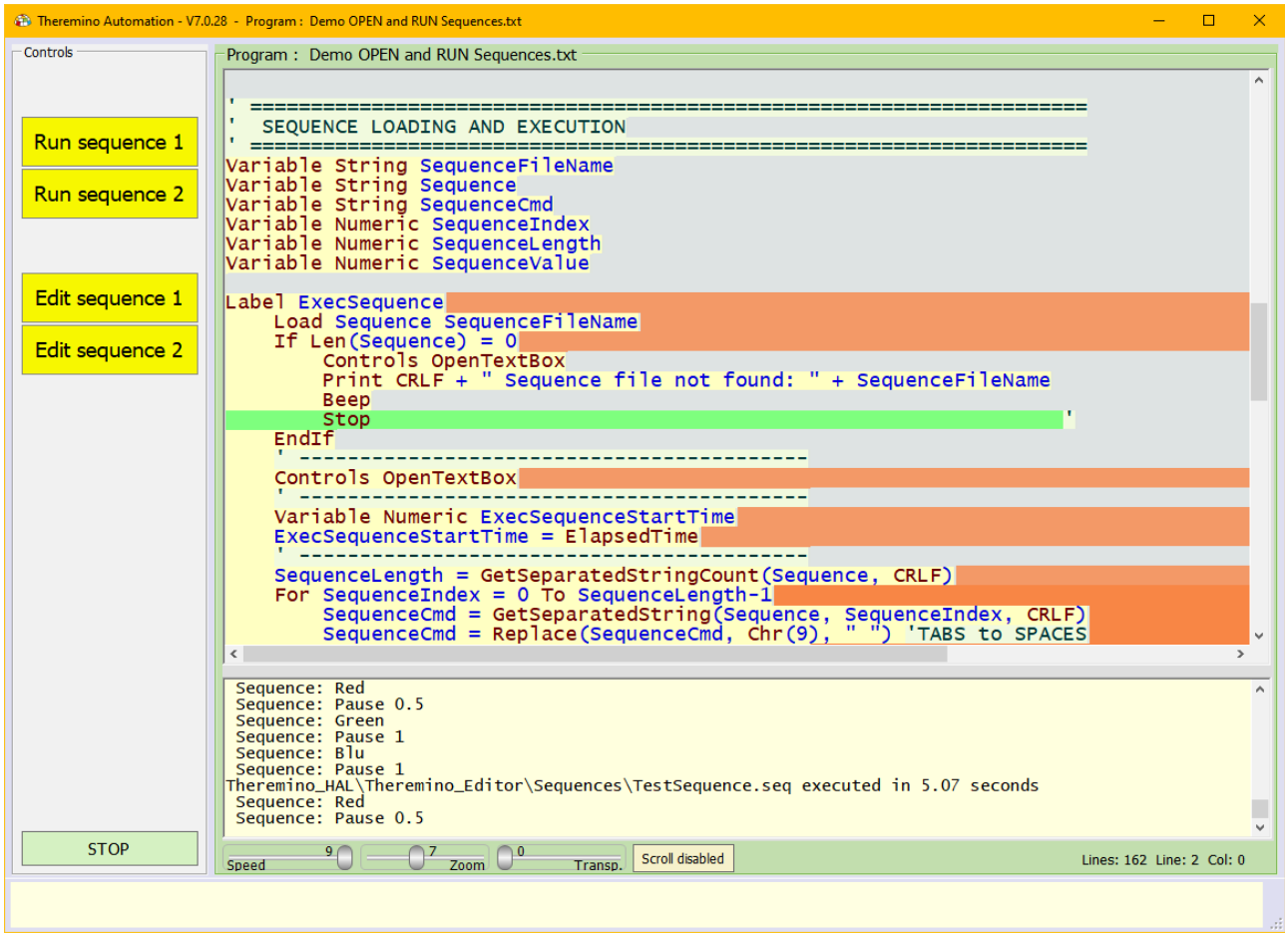
Sperimentate con l'esempio "Demo SEQUENCES"
che si trova nella cartella "Demo Programs \ Demo Sequences"

Utilizzare le "Sequenze"

Le sequenze sono solo un elenco di comandi scritto in un file di testo. Ma non è un linguaggio! Decidi tu le parole da eseguire, le scrivi una per riga, e poi nel programma di Automation scrivi quello che serve per eseguirle.

Nel programma di Automation le sequenze vengono eseguite da una apposita funzione che noi normalmente chiamiamo "ExecSequence". A questa funzione si devono aggiungere tutti i "Case" che servono per eseguire i comandi. Alcuni dei comandi possono anche includere dei parametri.

Usare le sequenze non è facile, bisogna decidere i nomi dei comandi, scrivere il necessario per eseguirli e anche aggiungerli alla lista dei comandi validi dell'editor delle sequenze. La lista dei comandi validi serve solo per cambiare colore in caso di errore, ma è un grande aiuto per non sbagliare.



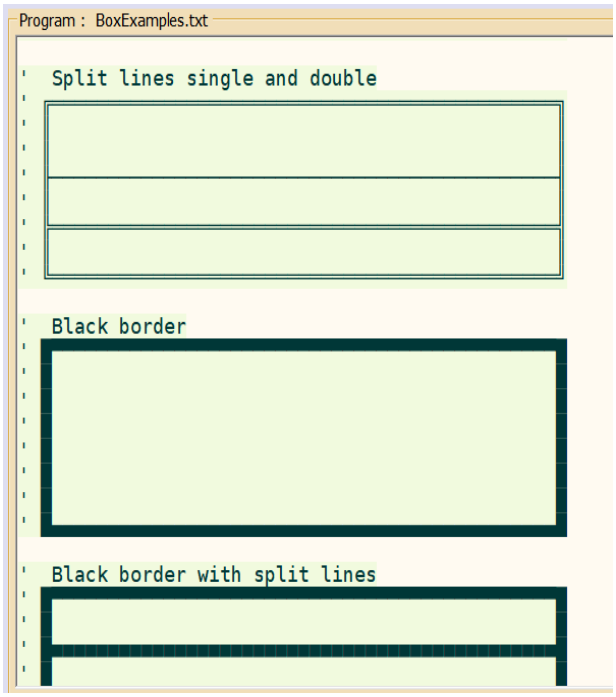
Sperimentate con l'esempio "Demo OPEN and RUN Sequences"
che si trova nella cartella "Demo Programs \ Demo Sequences"

Utilizzare i caratteri grafici

Con i caratteri grafici si possono costruire rettangoli e linee di separazione, utili per scrivere titoli e evidenziare le varie zone che compongono i programmi.

Per scrivere i caratteri grafici nei propri programmi li si copiano uno per uno dal file "GraphicChars.txt".

Per facilitare la composizione dei rettangoli abbiamo preparato un secondo file che si chiama "BoxExamples.txt", da cui copiare i rettangoli completi, per poi eventualmente modificarli, aggiungendo righe per allungarli o aggiungendo caratteri in mezzo per allargarli.



Ambedue i file si trovano nella cartella "Demo Programs" \ "Graphic Chars"

Dovendo copiare molti caratteri è possibile semplificare le operazioni di copia e incolla tenendo aperte due copie di Automation, una con i file di caratteri grafici da copiare e l'altra con il proprio programma su cui si incolleranno i caratteri. Per aprire una seconda copia di Automation si deve tenere premuto SHIFT mentre la si avvia.

Per visualizzare tutti i caratteri grafici, anche i più strani,
si devono scegliere il font "DejaVu Sans Mono" o il "Fira Mono".

Se ci si limita ai caratteri grafici più comuni, compresi i caratteri per le cornici, si possono utilizzare anche i font: "Courier New", "Cousine" e "Lucida Console".

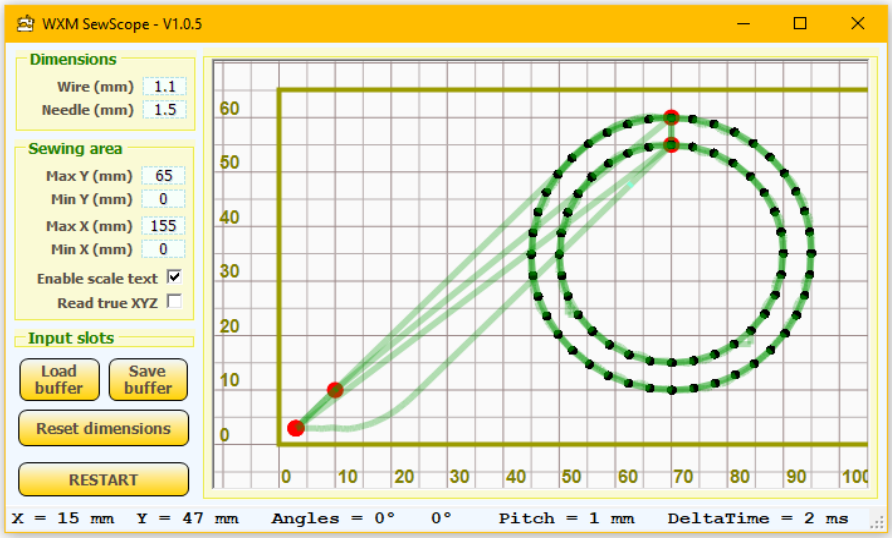
Funzioni per le cucitrici

Queste funzioni controllano la posizione della stoffa e il movimento dell'ago nelle macchine da cucire industriali (Sewing Machines).

La applicazione SewScope

Questa applicazione simula i movimenti di una macchina da cucire e può essere utile per mettere a punto i programmi di cucitura senza usare filo e stoffa.

La applicazione SewScope è sempre disponibile nella cartella Apps e può essere aperta dai programmi di automation con il comando Load.



Limiti di funzionamento per le funzioni Sew

Lo Stop_Angle è l'angolo dell'asse Z quando l'ago entra nel tessuto.
Lo Start_Angle è l'angolo dell'asse Z quando l'ago esce dal tessuto.

Quando l'asse Z ruota su Start_Angle, gl assi X e Y vengono spostati alla destinazione successiva con la massima accelerazione e velocità possibili.

Per compensare i limiti di accelerazione dell'asse Z e ridurre al minimo le imprecisioni ad alta velocità, l'impostazione Start_Angle deve essere aumentata da un valore standard di 270° a circa 350°

Lo StopAngle non viene utilizzato nelle funzioni "Sew", ma è l'angolo con cui l'ago entra nel tessuto. Se l'asse Z ruota oltre StopAngle, quando X e Y non sono ancora arrivati a destinazione, l'ago si sposta lateralmente e la cucitura diventa imprecisa. Questo limita la velocità lineare massima (mm/s) e la velocità massima di cucitura (punti/sec).

Sew functions

Queste funzioni (usate raramente e poco sperimentate) muovono i tre motori stepper delle macchine da cucire computerizzate.

I motori X e Y muovono la stoffa e il motore Z abbassa e alza l'ago ad ogni giro.

Nel motore Z il valore 1 rappresenta un giro di 360 gradi, per cui i multipli di 1 indicano "ago su" e i multipli di 0.5 indicano "ago giù".

- **`SewInit SlotX SlotY SlotZ StopAngle StartAngle`**

  Parametri di inizializzazione (tutti valori interi).
  I tre Slot sono un numero intero da 1 a 999.
  StopAngle è l'angolo dove fermare XY (circa 90 gradi - non usato).
  StartAngle è l'angolo dove iniziare a muovere XY (circa 350 gradi).

- **`SewLimits Xmin Xmax Ymin Ymax MaxPerSec`**

  Limiti di movimento (tutti valori float).
  Xmin, Xmax, Ymin e Ymax sono in millimetri.
  MaxPerSec è il massimo numero di punti al secondo.

- **`SewTo DestX DestY Speed Pitch Options`**

  Istruzione che avvia l'esecuzione di un segmento di cucitura.
  DestX e DestY sono le destinazioni di fine segmento in millimetri.
  Speed è la velocità di movimento in mm al secondo.
  Pitch è la larghezza dei punti di cucitura in millimetri.
  Options è una stringa di opzioni che inizia e finisce con doppi apici.

- **`SewAbort`**

  Il comando SewTo è asincrono e viene eseguito in un thread separato fino a che il segmento è completato o fino a che lo si interrompe con SewAbort.

- **`Var1 = SewProgress`**

  La variabile Var1 riceve un numero da 0 a 1.
  Questo numero indica quanta parte dell'ultimo SewTo è già stata eseguita.

Sew functions - Examples and Notes

Esempi di opzioni per la funzione SewTo

	DestX	DestY	Speed	Pitch	Options	
SewTo	70	35	20	4	"Radius=5"	Cerchio con raggio 5 mm
SewTo	70	35	20	2	"ZigZag=2"	ZigZag di 2 mm (+/- 1mm)
SewTo	10	30	20	5	"Begin=2"	Due Punti iniziali
SewTo	10	30	20	5	"End=3"	Tre punti finali

Le opzioni `"Begin" e "End"` non sono implementate.

Si possono anche combinare tutte le opzioni, come nell'esempio seguente

	DestX	DestY	Speed	Pitch	Options
SewTo	70	35	20	4	"Radius=5 ZigZag=2 Begin=2 End=3"

Limiti di velocità provati su una macchina da cucire con tre motori stepper di medie dimensioni

- Massima velocità lineare: 20 mm/s (imprecisioni visibili con 30 mm/s)
- Massima velocità di cucitura: 10 punti/s (imprecisioni visibili con 15 punti/s)

Regolazioni dei motori sulla applicazione HAL

- Motori X e Y --- MaxSpeed=15000 MaxAcc=5000 StepsPerMM=16
- Motore Z --- MaxSpeed=3000 MaxAcc=260 StepsPerMM=400

Tabella semplificata per la regolazioni del parametro di velocità

Pitch	Velocità massima	Velocità limite	Punti al secondo risultanti
1 mm	10 mm/s	15 mm/s	10 (15 max)
2 mm	20 mm/s	30 mm/s	10 (15 max)
>2 mm	20 mm/s	30 mm/s	meno di 10

Difetti conosciuti

Il singolo apice non funziona bene

Il singolo apice è quello che si usa per iniziare i commenti. Può accadere che premendo il singolo apice sulla tastiera questo non appaia. Per farlo apparire si deve premere il tasto una seconda volta.

Questo difetto è accaduto solo in Cina su alcuni computer che erano stati installati in modo particolare. Non siamo riusciti a riprodurlo per cui non possiamo correggerlo.

Per risolverlo si deve selezionare la tastiera "English US" invece della tastiera "English international". La selezione della tastiera è nella barra inferiore di Windows, a destra, vicino all'orologio.

Scritte troppo grandi sui pulsanti di controllo

In alcuni computer in Cina è accaduto che i pulsanti sulla sinistra della applicazione avessero scritte troppo grandi. Non siamo riusciti a riprodurre questo difetto.

Variabili erroneamente indicate come duplicate e altri errori

A volte alcune variabili si colorano di rosso e portandovi sopra il cursore si legge che sono duplicate oppure vengono segnalati altri errori. Spesso basta usare le frecce SU e GIU, o premere il pulsante "Select next error", e gli errori spariscono.

In caso di errori ostinati e incomprensibili si consiglia di premere il tasto destro del mouse sul pulsante LOAD. In questo modo il programma viene ricaricato e tutte le tabelle relative agli errori vengono svuotate e ricostruite. A volte con questo metodo alcuni errori si sistemano automaticamente.

Altri difetti conosciuti

Attualmente (Versione 7.8 del 1/Dicembre/2024), non conosciamo altri difetti. Se ne trovate scriveteci a: ***engineering@theremino.com***