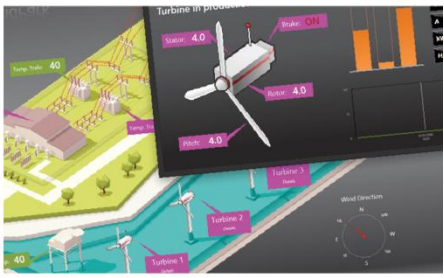
INDUSTRIAL AUTOMATION SOFTWARE
progea

Movicon.next™
Automation Platform.next generation



Movicon NExT

12.0 Basic Script

Ver.3.4.268

Sommario

1. Prefazione

1.1. Premesse

Tutte le informazioni contenute nella documentazione di Movicon.NExT presuppongono:

- Windows si riferisca all'apposito marchio registrato da Microsoft inc. nelle versioni Windows 7, Windows 8 e Windows 10 a 64 bit
- Platform.NExT si riferisca al sistema di supervisione sviluppato da Progea tutelato dalle leggi internazionali sul Copyright
- WWB.NET si riferisca al componente WinWrap.NET™ di Polar Engineering integrato in Platform.NExT, compatibile con il linguaggio Microsoft VB.NET.
- Ms Access e SQL Server si riferiscano ai prodotti registrati da Microsoft inc
- Ogni altro prodotto o marchio citato sia registrato o tutelato dal relativo proprietario

Attenzione: L'utilizzo di codice script all'interno di un progetto Movicon è sottoposto ad una GARANZIA LIMITATA. Infatti, l'Utente ha ampia libertà di gestire lo script per utilizzare eventi o metodi, sia di prodotto che di componenti di terze parti. Progea potrebbe non essere in grado di garantire completamente il corretto comportamento del prodotto in caso di codice script utilizzante l'esposizione di proprietà, eventi o metodi particolari. L'Utente deve sempre fare riferimento alla documentazione ove disponibile, ed eventualmente verificare preventivamente la corretta funzionalità del codice che intende utilizzare. In ogni caso Progea fornisce una GARANZIA LIMITATA sulle funzionalità di un progetto facenti uso di script VB.NET. In caso di necessità o di richiesta di ulteriori informazioni, contattare il Servizio Assistenza di Progea.

1.2. Introduzione

Contenuto della Guida alla programmazione dei Linguaggi Basic Script di Platform.NExT.

Il contenuto della Guida alla programmazione dei Linguaggi script Basic WWB.NET™ (compatibile Visual Basic.NET) contiene tutte le informazioni necessarie allo sviluppatore per la realizzazione di routines "Basic Script" di Movicon.NExT.
La guida comprende una parte relativa alle istruzioni inerenti ai comandi, metodi e funzioni API specifici per Movicon.NExT, ed una seconda parte contenente la sintassi delle istruzioni standard compatibile WWB.NET™, facente parte della guida **"WinWrap Basic Language"** (Copyright Polar Engineering) incorporata in Movicon.NExT™.

2. Concetti Generali

2.1. Proprietà Basic Script

2.1.1. Proprietà di un Basic Script

Ogni routine **"Basic Script"** inserita come risorsa nella finestra "Esploratore Progetto" può ricevere in associazione delle Proprietà, che ne determinano il tipo di funzionamento durante l'esecuzione. Per fare questo è sufficiente selezionare il "Basic Script" desiderato e quindi modificare le impostazioni tramite la "**Finestra delle Proprietà**" di Movicon.NExT.

Generale

Tag Tempo Ciclo

In questo campo è possibile associare un Tag sul quale verrà riportato il valore del tempo di ciclo della risorsa Basic Script. Il valore è espresso in millisecondi e verrà aggiornato al termine dell'esecuzione dello Script. Ne risulta pertanto che se lo Script ha al suo interno un Loop infinito il valore di tempo ciclo resterà a zero.

Per questioni di performance l'avvio dello Script non attende che il Tag di "Tempo Ciclo" sia stato sottoscritto al Server. Questo può comportare che il Tag di "Tempo Ciclo" può iniziare ad essere aggiornato dopo che lo Script è già in esecuzione.

Tag Stato

In questo campo è possibile associare un Tag sul quale verrà riportato lo stato di esecuzione dello Script. Gli stati dello Script sono gestiti a bit, ovvero i singoli bit del Tag hanno il seguente significato:

Bit 0: non usato.
Bit 1: Error. Il bit va a True ogni volta che lo Script è in errore.
Bit 2: non usato.
Bit 3: non usato.
Bit 4: DoEventing. Il bit va a True ogni volta che si esegue un'istruzione DoEvent all'interno dello Script.
Bit 5: Starting. Il bit va a True quando lo Script è in fase di avvio.
Bit 6: Stopping. Il bit va a True quando lo Script è in fase di arresto.
Bit 7: Running. Il bit va a True quando lo Script è in esecuzione.

per questioni di performance l'avvio dello Script non attende che il Tag di "Stato" sia stato sottoscritto al Server. Questo può comportare che il Tag di "Stato" può iniziare ad essere aggiornato dopo che lo Script è già in esecuzione.

Priorità Thread

L'esecuzione di una risorsa basic script può essere eseguito con diverse priorità: alta, normale e bassa:

Lowest: priorità molto bassa

Below Normal: priorità inferiore a Normal
Normal: Priorità normale
Above Normal: priorità superiore a Normal
Highest: massima priorità

Force Writing On Server
Proprietà di dafault a false, se impostata a true fa sì che lo script scriva il valore in una variabile incondizionatamente da quello precedente, quindi permette di riscrivere lo stesso valore all'interno di una tag.

Esecuzione

Timeout Arresto
Tempo di timeout per l'arresto di uno Script.

Tempo di attesa
Questo tempo di sleep, espresso in millisecondi, serve per rendere meno onerosa l'esecuzione del basic script per il processore. Più alto sarà il tempo di sleep più il basic script impegnerà meno il processore e di conseguenza rallenterà la sua esecuzione.

Impostazioni di Connessione

Nome Sessione
Specificando un nome in questo campo lo Script verrà eseguita in una sessione di connessione al Server differente da quella del Client Movicon. Se lo Script gira in una sessione differente significa che i parametri di connessione al Server saranno quelli definiti nella risorsa Script, che vengono riportati a seguito. Se invece lo Script viene fatto girare nella stessa sessione del Client Movicon, quindi il parametro "Nome Sessione" viene lasciato vuoto, allora i parametri di connessione al Server saranno gli stessi del Client Movicon, quelli definiti nelle proprietà "Impostazioni di Connessione" del progetto.

Ritardo rimozione Item
Parametro gestito solo se è impostata la Proprietà "Nome Sessione" dello script. Imposta il tempo di ritardo con il quale verranno rimosse le sottoscrizioni OPC UA degli ITEM non attivi espresso in secondi.

Numero Item da Rimuovere
Parametro gestito solo se è impostata la Proprietà "Nome Sessione" dello script. Imposta il numero di Item OPC non attivi da rimuovere ad ogni intervallo di tempo.

Usa Connessioni Sicure
Parametro gestito solo se è impostata la Proprietà "Nome Sessione" dello script. Abilitando questa opzione verranno consentite soltanto le connessioni al Server che risultano "sicure".

Tempo di aggiornamento Item Attivi
Parametro gestito solo se è impostata la Proprietà "Nome Sessione" delo script. Definisce la frequenza d'aggiornamento per una variabile in uso ed esistente. Il parametro viene passato al Server, e al Driver eventuale, quando si carica lo script e le variabili entrano in uso. Nel caso in cui non sia definita una Sessione per lo scipt, viene utilizzata quella definita a livello di progetto e di conseguenza anche il valore generale del parametro "Tempo di Aggiornamento Item Attivi".

Tempo di aggiornamento Item non Attivi

Parametro gestito solo se è impostata la Proprietà "Nome Sessione" dello script. Definisce la frequenza d'aggiornamento per una variabile che sta per andare non in uso ma esistente. Il parametro viene passato al Server, e al Driver eventuale, quando si scarica lo script. Nel caso in cui non sia definita una Sessione per lo script, viene utilizzata quella definita a livello di progetto e di conseguenza anche il valore generale del parametro "Tempo di aggiornamento Item non Attivi".

Disabilita se non in uso

Imposta i Tag come "Inactive" quando non sono in uso.

Tempo di aggiornamento

Tempo di notifica dei Tag verso il Server.

Debug

Enable Log

Abilitando questa casella di selezione, durante la fase di runtime verrà creato un file xml nella cartella dello script per la routine Basic Script, dove verranno stampati i messaggi inerenti allo script, come ad esempio i Debug.Print. Il file rimarrà in uso fino a quando lo script non verrà fermato per questioni di performance.
Al riavvio il file verrà comunque sempre rigenerato.

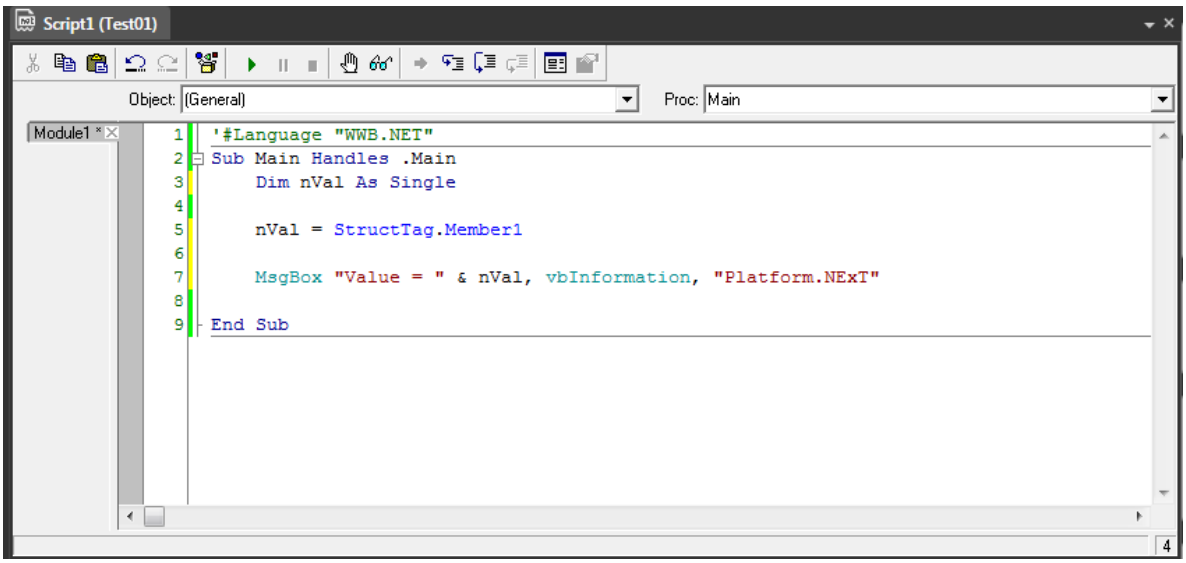
Enable SysLog

Abilitando questa proprietà i messaggi eseguiti dalla funzione Debug.Print verranno stampati anche nel Log Storico del progetto.

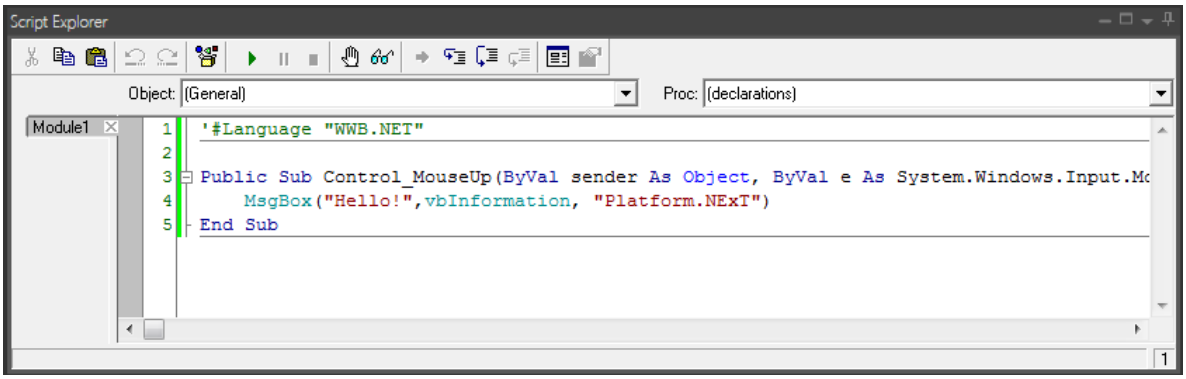
2.2. Editor Basic Script

2.2.1. Editor Basic Script

L'editor basic script si compone di una finestra comprendente un'area con sfondo bianco in cui è possibile editare il codice, più una serie di comandi descritti di seguito. La dimensione della finestra può essere ridimensionata e la possibilità di aprire contemporaneamente più finestre permette di eseguire rapidamente operazioni di Copia-Incolla anche da progetti diversi. Per aprire l'editor è sufficiente fare un doppio click con il mouse sulla risorsa Basic Script.



L'editazione del codice script degli oggetti e dei sinottici invece viene fatta tramite la finestra **"Esploratore Script"** di Platform.NExT. Il contenuto di questa finestra infatti cambia dinamicamente in base al componente selezionato, mostrando il codice script ad esso associato.



Entrambe le finestre di editing sopra mostrate mettono a disposizione due liste a discesa, **"Object:"** e **"Proc:"**.

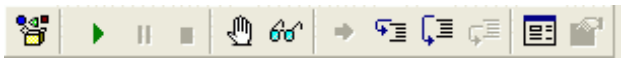
Nella lista **"Object:"** è visualizzato l'elenco degli oggetti basic definiti in Movicon.NExT che corrispondono a **"(General)", "Control", "Document", "Entity" e "Parent"**. La selezione di un oggetto rispetto ad un altro modifica l'elenco delle procedure nella lista **"Proc:"** disponibili per quello specifico oggetto.

"(General)" : identifica le procedure programmate per quello specifico oggetto e l'elenco può cambiare a seconda dell'oggetto selezionato sul sinottico.
"Document": è un'interfaccia che fà riferimento al sinottico. Se ad esempio si usa l'evento "VariableChanged" verranno notificati i cambiamenti delle variabili all'interno del sinottico.
"Control": identifica gli eventi di un oggetto dello screen ed è disponibile solo quando si edita il codice associato a un oggetto del sinottico.
"Entity": identifica il controllo come oggetto del framework.
"Parent": è un'iterfaccia che fa riferimento al progetto.

Nella lista **"Proc:"** è visualizzato l'elenco degli eventi disponibili in base alla seleziona fatta nella lista "Object:".

2.2.2. Barra Utensili Basic Script

Aprendo la finestre di editazione del codice basic script, di una risorsa o dall'"Esploratore Script", sulla parte superiore viene mostrata la barra utensile Basic Script.



Alcuni comandi sono eseguibili anche dal "Menù Debug".
La funzione dei tasti è di seguito elencata:

Browse

Questo comando consente di accedere all'elenco delle funzioni e proprietà disponibili nel basic script. La finestra che si apre è chiamata Browse delle funzioni.

Start/Resume

Questo comando avvia l'esecuzione del basic script. Se il codice è riferito ad una risorsa basic del progetto viene eseguito il contenuto della procedura Sub Main altrimenti se è riferito ad un oggetto abilita la gestione degli eventi su di esso. Viene usato per fare un controllo sintattico del codice scritto, in caso non si faccia riferimento a risorse del progetto, che richiedono l'esecuzione in run del progetto stesso (es lettura di variabili), viene anche eseguito il codice stesso (es lettura di un valore da DB).

Pause

Questo comando mette in pausa il basic script che è in esecuzione posizionandosi e colorando di giallo la linea di codice che era in esecuzione.

End

Questo comando arresta l'esecuzione del basic script. Il codice deve essere in pausa per agire su questo comando.

Break Point

Questo comando inserisce o elimina un break point sulla riga in cui è presente il cursore. Questa funzione è disponibile anche con un click sull'estremo bordo sinistro della finestra di editor in coincidenza della riga nella quale si vuole inserire il break point o con il tasto F9. Nel momento in cui il basic deve eseguire una riga di codice in cui è presente un break point l'esecuzione viene messa automaticamente in pausa.

Quick Watch

Questo comando restituisce su una finestra di dialogo o nella finestra di debug (finestra di Watch) del basic il risultato della funzione che è stata evidenziata.

Show Current Statement

Questo comando consente di posizionare il cursore all'altezza della prossima istruzione che dovrà essere eseguita durante la fase di debug della routine.

Step Into

Questo comando esegue passo-passo una funzione alla volta ad ogni pressione del comando.

Step Over

Questo comando esegue passo-passo una funzione alla volta ad ogni pressione del comando.

Step Out

Questo comando esegue le funzioni presenti fino ad arrivare nella linea in cui è presente il cursore.

Edit UserDialog

Questo comando apre il tool per la creazione grafica delle finestre di dialogo basic script. All'uscita con conferma viene tradotto ciò che è stato creato graficamente in codice. Posizionandosi su questo e attivando il comando, viene nuovamente riprodotto graficamente ciò che è espresso in codice basic script.

Edit Module Propreties

2.2.3. Debug dei Basic Script

Movicon.NExT consente il completo debug delle logiche basic script del progetto, siano esse risorse del progetto o contenute nei simboli grafici dei sinottici.
Nella fase di progettazione è possibile debuggare solo i Basic Script definite come Risorse, mentre non è possibile testare gli script associati agli oggetti in quanto questi sono gestiti ad evento. Durante la fase di Runt-Time invece è possibile debuggare tutti gli script. In fase di programmazione è necessario inserire dei Break-Point all'interno del codice. Quando in Run-Time lo script verrà eseguito si aprirà la finestra di debug e l'esecuzione del codice sarà ferma alla posizione del Break-Point. A questo punto tramite la barra di debug posta in alto alla finestra sarà possibile eseguire lo script step a step, ecc.

L'inserimento di Break-Point all'interno degli script è permanente, ovvero vengono salvati nelle proprietà dello script, e quindi saranno attivi anche in modalità Run-Time. Si raccomanda quindi di inserire i Break-Point solo durante la fase di debug e di rimuoverli subito dopo avere eseguito i test.

Alcune funzioni basic script verificate in programmazione possono ritornare con valori diversi se eseguite in runtime, quindi si raccomanda di utilizzare il debug in programmazione solamente per verificare in maniera grossolana il codice e ripetere il test in maniera accurata lanciando in esecuzione il progetto.

2.3. Basic Script nel Progetto

2.3.1. WinWrap Basic Language

Movicon.NExT integra al suo interno un componente software, WinWrap Basic, che permette di editare all'interno dell'applicazione delle routine in linguaggio Basic.
Di seguito vengono elencate alcune delle principali caratteristiche del WinWrap Basic:

- permette di creare delle routine in linguaggio WWB.NET compatibili con il codice BASIC (Beginners All-purpose Symbolic Instruction Code)
- permette di estendere il set di istruzioni con funzioni e metodi personalizzati
- permette di creare delle dialog box utilizzando i controlli standard di windows (pulsanti, checkboxes, groupboxes, listboxes, option buttons, immagini e caselle di testo)
- Consente di inserire dei riferimenti agli assembly installati con il framework 4.5.2 o sviluppati a partire dal framework.

Supporto al VB.NET

Il linguaggio predefinito è il WWB.NET, questo viene abilitato tramite la direttiva '#Language "WWB.NET" inserita di default all'inizio di una risorsa script. Questo consente di accedere direttamente alle .NET assemblies con il codice VB. La finestra di dialogo per aggiungere gli oggetti di riferimento da usare nel codice propone l'elenco di tutti gli assemblies selezionabili.

Esempio di codice che presuppone di avere aggiunto un riferimento all'assembly System.Windows.Forms:

```
'#Reference #System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089, processorArchitecture=MSIL

'#Language "WWB.NET"

Dim WithEvents t As System.Windows.Forms.Timer

Sub Main
    t = New System.Windows.Forms.Timer
    t.Interval = 1000
    t.Enabled = True
    Wait 1
End Sub

Private Sub t_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
t.Tick
    Debug.Print Now
End Sub
```

A tale scopo sono disponibili tutta una serie di istruzioni nuove per rendere il codice compatibile alla programmazione VB.NET.

Limitazioni:

- Le firme delle routine negli oggetti non possono essere rinominate così come consentirebbe la programmazione .NET

- Non è possibile collegare più eventi alla stessa procedure usando una lista di "Handles"
- Non è possibile gestire eventi sulla modifica del valore di una variabile del progetto in uno script WWB.NET
- Nella versione 9 del WinWrap Basic, il basic script è single thread e il codice script deve essere eseguito dal thread che l'ha creato. Ad esempio non è possibile passare un dalegate di .NET ad un oggetto che gestirà la chiamata di questo delegate in un altro thread. Un esempio concreto sono alcune delle funzioni della classe "System.IO.FileSystemWatcher" che creano dei thread aggiuntivi per monitorare le modifiche dei file. In questo caso i metodi di tale classe non sono supportati in quanto richiamano altri thread. La stessa cosa vale anche per la classe "System.Net.Sockets"
- Nella versione 9 del WinWrap Basic, il basic script non supporta i "nested types", cioè un tipo definito nell'ambito di un altro tipo. Ad esempio non è possibile usare le funzioni del tipo "System.Net.WebRequestMethods.Ftp".

Supporto Unicode

L'editor del codice consente di inserire delle stringhe in formato Unicode. Quindi è possibile visualizzare nelle finestre di dialogo basic script delle stringhe Unicode oppure assegnare a delle variabili stringa del progetto testi Unicode.
Esiste la possibilità di leggere e/o scrivere file Unicode UTF8 o UTF16. A tale scopo sono state aggiunte due nuove costanti "vbUTF8BOM" e "vbUTF16BOM". Scrivendo una di questa costanti nel primo carattere di un file di testo ne verrà determinata la codifica.

2.3.2. Basic Script come Risorsa

Se si desidera inserire un Basic Script nel progetto, occorre procedere all'inserimento di una nuova risorsa. Per l'inserimento di una nuova risorsa Basic Script, selezionare prima il punto desiderato nella struttura ad albero del gruppo "Scripts" della finestra "Esploratore Progetto", quindi cliccare il pulsante "New" dal menù contestuale che verrà aperto.
Alla conferma dell'operazione, apparirà nel gruppo o nel punto selezionato nella struttura delle Risorse, la nuova risorsa Basic Script, e verrà aperta la finestra di editor del codice relativa al Basic Script appena inserito. A questo punto è possibile procedere all'introduzione del codice VB.NET™ comp. come descritto nei paragrafi specifici.

Alla risorsa potrà successivamente essere assegnato il Nome utilizzando il mouse facendo clic sulla risorsa e digitando il nome in sostituzione a quello provvisorio.

Una risorsa Basic Script deve contenere al suo interno la procedura Main (Sub Main). Le istruzioni contenute in tale subroutine vengono eseguite nel momento in cui il basic è lanciato dalle logiche del progetto. Al termine della subroutine, in assenza di cicli loop programmati, il basic viene terminato ed è pronto per un successivo richiamo.
La procedura Main non presenta dei parametri configurabili ma è possibile associare un parametro al momento della chiamata del Basic Script. Nel comando che esegue il richiamo del basic script dovrà essere indicato il parametro che a sua volta può contenere diversi argomenti separati da un carattere separatore e all'interno del basic script la funzione StartupContext.Parameter() potrà essere utilizzata per leggere il valore del parametro con cui il basic è stato richiamato. Con la funzione vb.net “split” sarà poi possibile andare a leggere i singoli argomenti passando il carattere usato come separatore.

Dopo che una risorsa Basic Script è stata messa in esecuzione la prima volta, anche dopo che la Sub Main() è terminata, la risorsa rimane attiva, quindi eventuali variabili riferite al suo interno saranno conteggiate ai fini dalla licenza. Se si vuole invece terminare completamente l'esecuzione di un Basic Script è necessario eseguire il comando di "Stop" disponibile ad esempio nella "Lista Comandi" di Movicon.NExT**.**

Il comando di stop di una risorsa basic script scarica dalla memoria soltanto quelle risorse basic che sono in esecuzione in un thread separato. Le altre risorse basic script che sono in esecuzione nello stesso thread vengono solo fermate. Ne consegue che l'evento "Unloading" non viene più eseguito a seguito di un comando di stop per quelle risorse basic script che non sono in thread separato. Inoltre le variabili utilizzate dal basic script rimangono sempre in uso una volta che il basic script è stato eseguito almeno una volta.

Una risorsa Basic Script introdotta nel progetto può essere mandata in esecuzione in vari modi, in funzione delle necessità. I paragrafi seguenti ne illustrano le metodologie.

Esecuzione su comando

L'esecuzione su comando è facilmente impostabile tramite la lista "Comandi" degli oggetti, dei menù o della risorse che prevedono la possibilità di eseguire dei comandi. Ad esempio, se ad un pulsante si desidera associare l'esecuzione di una routine Basic Script, occorre selezionare dalla proprietà "Edita proprietà comuni"->"Comandi"->"Script", selezionando la risorsa Basic desiderata tra la lista proposta riferita a quelle introdotte, e definendo poi le altre impostazioni come desiderato.
L'esecuzione su comando di un Basic Script può essere eseguita anche dalla risorsa "Gestore Eventi", quindi di fatto l'avvio della routine basic non verrà gestita da un comando da operatore, ma sarà eseguita a fronte di un determinato evento.

Nel caso di comando script le modalità d'invocazione sono le seguenti:

- Normal: è il valore predefinito, lo script viene avviato in modalità asincrona rispetto alla chiamata.
- Synchro: lo script viene avviato in modalità sincrona, la chiamata attende il completamento dell'esecuzione prima di riprendere il controllo (in questa modalità il dubug è disabilitato).
- Shared: Lo script condivide lo stesso thread di esecuzione di altri script che sono stati avviati nella stessa modalità. Questo consente di utilizzare un numero più limitato di risorse della macchina.
- Stop: lo script viene scaricato dalla memoria.

Esecuzione allo startup

E' possibile eseguire una routine Basic Script automaticamente all'avvio in runtime del progetto applicativo. Per utilizzare questa funzionalità, occorre accedere all'apposita impostazione "Startup Script List" dalle proprietà del progetto.
Attraverso l'apposita finestra di selezione, occorre selezionare la risorsa Basic Script tra quelle precedentemente inserite nel progetto.

2.3.3. Basic Script nel Progetto

All'interno di un progetto è possibile disporre delle funzioni basic script in diverse circostanze e modalità.
E' bene farne utilizzo in quei frangenti in cui le stesse operazioni non siano disponibili con altre risorse o metodi: proprietà di esecuzione/animazione dei simboli, Eventi, Schedulatori, ecc.. L'utilizzo incondizionato dei basic script all'interno di un progetto potrà essere molto comodo in fase di progettazione ma potrà rendere meno veloce e più dispendiosa di risorse l'esecuzione del progetto.
Il codice basic può essere utilizzato in più punti del progetto: come risorsa, direttamente nelle proprietà di esecuzione di un simbolo, come codice associato agli eventi di un oggetto (allarme, disegno o simbolo), nei sinottici.
Nelle apposite sezioni sono spiegati in dettaglio queste funzionalità.

2.3.4. Gestore eventi

Alcune classi del framework utilizzabili all'interno dell'editor script, consentono di gestire degli eventi.vediamo di seguito alcuni esempi:

la classe **Timer:**

```
'#Language "WWB.NET"
Imports System.Windows.Forms
Dim WithEvents t As System.Windows.Forms.Timer
Sub Main
    t = New System.Windows.Forms.Timer
    'Preset Timer
    t.Interval = 1000
    'Start/Stop Timer
    If t.Enabled Then
        t.Enabled = False
    Else
        t.Enabled = True
    End If
End Sub
    'Questo evento scatta al termine di ogni intervallo impostato
Private Sub t_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles t.Tick
   ScriptDocument.SetVariableValue("ActualTime", Format(Now, "dddd dd-mm-yyyy") &
vbLf & Format(Now, "hh:nn:ss"))
End Sub
```

Un altro esempio è la classe **FileSystemWatcher** gestisce un evento che scatta ogni volta viene modificato un file monitorato dalla classe stessa.

```
'#Language "WWB.NET"
Imports System.IO
Sub Main Handles .Main
Try
     Dim  path As String= "c:\Test"
    'WatchTest.Log è il file che viene monitorato
    Dim watch As New FileSystemWatcher(path, "WatchTest.Log")
    'Aggiungo un gestore evento per l'evento previsto "Changed"
    AddHandler watch.Changed, AddressOf watch_Changed
```

```
    watch.EnableRaisingEvents = True
    'il loop mantiene attivo lo script e consente di fare scattare l'evento
    'che nel caso della FileSystemWatcher viene avviato da un thread distinto dal
chiamante
    Do
        DoEvents
    Loop Until ScriptDocument.IsInStoppingMode
Catch ex As System.Exception
    Debug.Print ex.Message
Finally
End Try
End Sub
Private Sub watch_Changed(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim actualVal As Integer
    'VariableChange sarà la variabile che conteggia il numero di cambiamenti
    actualVal= ScriptDocument.GetVariableValue("VariableChange")
    ScriptDocument.setVariableValue("VariableChange",actualVal+1)
End Sub
```

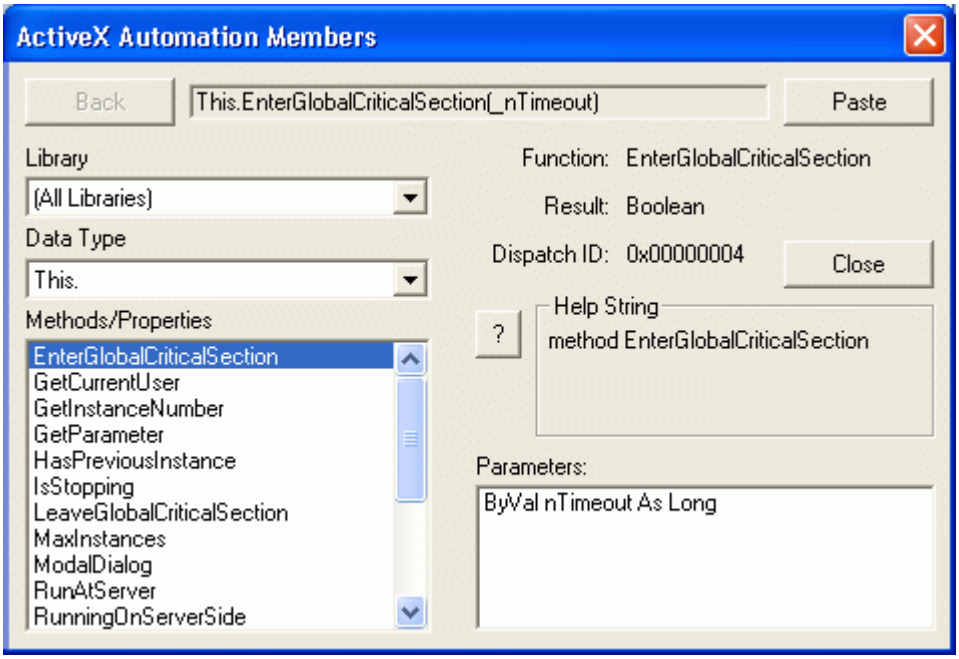
2.3.5. Codice nei Basic Script e nei Simboli

Esiste una fondamentale differenza di funzionamento tra le risorse Basic Script e il codice che si inserisce all'interno dei simboli, dei sinottici e degli allarmi. Nei Basic Script deve esistere la **"Sub Main"** che viene eseguita al richiamo del Basic Script. Nel caso invece dei simboli (o sinottici o allarmi) non esiste una Sub Main, ma la gestione del codice è fatta esclusivamente in base agli eventi (SymbolLoading, Click, ecc.) disponibili all'interno del codice e che sono stati inseriti dal programmatore. Movicon.NExT richiamerà tali eventi soltanto finché il simbolo (o sinottico) saranno attivi, quindi quando il sinottico è caricato in ram.

2.3.6. Librerie Basic Script

Oltre alle funzioni base messe a disposizione dal WinWrap Basic, all'interno dei Basic Script è possibile utilizzare una serie di funzioni supplementari che fanno parte delle librerie di Movicon.NExT e che permettono di interagire con il progetto. Queste funzioni permettono per esempio di leggere e scrivere le variabili dell'Address Space del Platform.NExT, di eseguire dei cambi pagina, di interagire con le proprietà dei simboli di Movicon.NExT, ecc. Queste librerie di funzioni di Movicon.NExT vengono definite come Librerie di Interfaccia Basic, e ogni interfaccia raccoglie una serie di funzioni specifiche per un determinato componente del progetto.
Le interfacce basic di Movicon.NExT sono numerose, e per accedere all'elenco delle funzioni di queste librerie si può utilizzare il pulsante "Browse" delle funzioni della barra utensili dei basic (disponibile dopo avere aperto il codice basic di un qualsiasi componente):

La finestra che si apre riporta il nome dell'interfaccia nella casella "Data Type" e la lista dei metodi e proprietà relativi all'interfaccia nella lista "Methods/Propreties":



Finestra di Browse delle funzioni basic script.

E' possibile selezionare un'altra interfaccia agendo sulla lista "Data Type".
Per ottenere l'help di una funzione è sufficiente selezionare la funzione e premere il punto interrogativo se ci si trova nella finestra del Browse delle funzioni, oppure se si è all'interno del codice basic è sufficiente selezionare la funzione e premere il tasto F1. Il significato dei vari campi della Finestra di Browse è il seguente:

Back
Ritorna di un passo indietro nella scala gerarchica della libreria. Alcune librerie definiscono degli oggetti ai quali sono associate delle proprietà e dei metodi, se stiamo visualizzando le proprietà di un oggetto con un clic sul pulsante back si ritornerebbe all'elenco degli oggetti definiti nella libreria.

Paste
Copia il contenuto della casella a lato, nella posizione in cui si trova il cursore sulla finestra di editor dello Script. Se il comando è disabilitato indica l'impossibilità di copiare il contenuto nella posizione in cui il cursore si trova.

Close
Chiude la finestra di browse e ritorna attivo l'editor dello Script.

Library
Permette di selezionare una delle librerie dall'elenco proposto. Se si aggiungono ActiveX/OCX attraverso i references questi sono disponibili nell'elenco.

Data Type
Permette di selezionare un tipo di dato dall'elenco proposto. L'elenco si riferisce alla libreria selezionata nella casella Library.

Methods/Properties
Permette di selezionare un metodo o una proprietà dall'elenco proposto. L'elenco si riferisce al tipo di dato selezionato nella casella Data Type.

?
Accede all'help della proprietà o del metodo selezionato nella casella a fianco. Alcune librerie esterne, non definite da Progea e abilitate attraverso i references, non installano i loro file di help.

Parameters
Viene visualizzato l'eventuale elenco di parametri definiti per il metodo selezionato nella casella Data Type.

2.3.7. Sub, Function, Eventi, Metodi e Proprietà

Sub e Function

All'interno dei basic script si possono creare delle Routine, ovvero porzioni di codice racchiuse in un blocco, che possono essere fondamentalmente di due tipi:

- Sub
- Function

la differenza tra questi due tipi di routine è in realtà minima. Entrambe possono essere richiamate passando loro dei parametri. La "Function" però a differenza della "Sub" può restituire un valore di tipo definito dal programmatore (Bool, Int, String, ecc.)

Esempio 1:
All'interno della Sub Main di un Basic Script viene richiamata la Sub Prova:

```
Sub Main
    Call Prova()
End Sub

Sub Prova()
    MsgBox("Prova Sub", vbInformation + vbOkOnly,GetProjectTitle)
End Sub
```

Esempio 2:
All'interno della Sub Main di un Basic Script viene richiamata la Function Prova e viene stampato il risultato del tasto premuto dall'utente sulla MsgBox:

```
Sub Main
    Debug.Print Prova()
End Sub

Function Prova() As String
    If MsgBox("Prova Function",vbInformation + vbOkCancel,GetProjectTitle) =
    vbOK Then
        Prova = "OK"
    Else
```

```
            Prova = "Annulla"
        End If
End Function
```

Le routine sopra descritte possono essere richiamate da altre routine, quindi si creano delle chiamate annidate tra loro. Nel caso della Risorsa Basic Script però dovrà sempre esistere la **"Sub Main"** che è la routine che viene eseguita automaticamente dal Basic Script quando questo viene messo in esecuzione. Sarà poi cura del programmatore inserire il codice opportuno all'interno di questa routine ed eventualmente richiamare altre Sub o Function create dal programmatore stesso. Una volta che le istruzioni contenute all'interno della Sub Main() sono state tutte eseguite il basic script viene terminato e si dovrà richiamarlo nuovamente per eseguirlo un'altra volta. E' possibile anche inserire dei loop all'interno della Sub Main() in modo da mantenere il Basic Script sempre in esecuzione.

Clausola Uses

All'interno degli script è possibile referenziare delle funzioni all'interno di altri script. Per fare questo è necessario utilizzare la clausola #Uses dentro lo script all'interno del quale si vogliono referenziare funzioni appartenenti ad altri script risorsa.

Un esempio di sintassi della clausola #Uses è il seguente:
'#Uses "*General\PublicFunctions"
Utilizzando questa sintassi è possibile, quindi, fare riferimento alle funzioni contenute nello script risorsa 'PublicFunctions' contenuto all'interno di una folder chiamata 'General'.

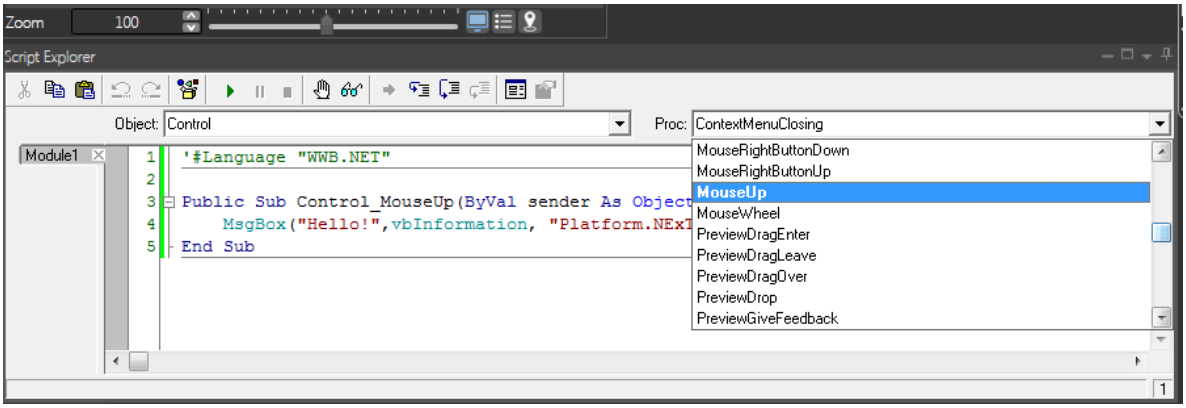
Nel caso in cui lo script chiamante risieda anch'esso all'interno di una folder differente, la sintassi della clausola #Uses diventa la seguente:

'#Uses "*..\General\PublicFunctions"

dove ..\ indica la root.

Eventi

Esistono delle routine di tipo Sub, messe a disposizione del sistema (Movicon.NExT o eventuali componenti ActiveX inseriti) che vengono richiamate automaticamente dal sistema a fronte di determinati eventi. Queste routine, che vengono appunto definite "Eventi", possono essere inserite negli script e sarà cura del programmatore aggiungere al loro interno il codice desiderato.



Esempio:
Inserendo il seguente codice all'interno dello script di un disegno rettangolo, ogni volta che si eseguirà un Click col mouse sul rettangolo apparirà una MsgBox con la scritta "Hello!":

```
'#Language "WWB.NET"
Public Sub Control_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseButtonEventArgs) Handles Control.MouseUp
        MsgBox("Hello!",vbInformation, "Platform.NExT")
End Sub
```

In questo caso la Public Sub Control_MouseUp() è un evento scatenato dal sistema quando si esegue un Mouse Up sul rettangolo, mentre il codice al suo interno deve essere inserito dal programmatore.

All'interno del codice basic dei simboli, degli oggetti di sinottico si possono selezionare una serie di eventi resi disponibili dal sistema. Questi eventi riguardano esclusivamente il componente sul quale si sta editando il codice, ovvero gli eventi in questione riguardano eventi scatenati dal componente. Questi eventi sono elencati nella lista "Proc:" quando è selezionata la voce "(Control)" nella lista "Object", come mostrato nella figura sopra.

Metodi e Proprietà

Le funzioni che si possono inserire dalle interfacce basic di Movicon.NExT possono essere esposte come "Metodi" e "Proprietà". Le proprietà identificano una caratteristica dell'oggetto in questione, ad esempio il colore, il titolo, ecc., mentre il metodo esegue una funzione di quell'oggetto.

2.3.8. Variabili nei Basic Script

All'interno dei Basic Script di Movicon.NExT possono essere utilizzate sia le variabili dell'Address Space di Movicon.NExT, che risulteranno essere comunque variabili globali per tutto il progetto, sia le variabili locali del Basic Script che possono essere visibili solo all'interno della routine Basic e che vengono distrutte una volta terminato il Basic Script. E' possibile inoltre utilizzare anche le variabili "Locali" e di "Sistema"del Client.
Le variabili del progetto (Address Space di Platform.NExT) possono essere utilizzate direttamente con il loro nome oppure utilizzando le apposite funzioni della "ScriptDocument.GetVariableValue()", "ScriptDocument .SetVariableValue()", nel caso di una risorsa Script o con le funzioni corrispondenti "Document.GetVariableValue()", "Document .SetVariableValue()" per il codice editato a livello degli oggetti di un sinottico.

Le variabili interne del Basic Script hanno priorità su quelle del progetto. Quindi utilizzando il nome di una variabile di progetto direttamente nel codice basic script, se esiste una variabile interna al basic con lo stesso nome, viene impostata o letta questa e non quella di progetto. In questo caso occorre appoggiarsi alle funzioni basic GetVariableValue(), SetVariableValue().

Le variabili di tipo bit, quando sono utilizzate direttamente con il loro nome, sono convertite in boolean (true o false). Questo significa che all'interno del Basic Script una variabile del progetto che ha valore "1" verrà letta come variabile "true" e quindi con un valore numerico uguale a "-1". Per quanto riguarda il valore "0" che viene interpretato dal Basic Script come "false" risulta essere sempre un valore numerico uguale a "0". Questo meccanismo è valido sia per la lettura che per la scrittura delle variabili di tipo bit all'interno dei Basic Script. Per ovviare a questo meccanismo è sempre possibile utilizzare le funzioni basic GetVariableValue() e SetVariableValue() per leggere e scrivere le variabili di tipo bit. In questo modo il valore di ritorno della funzione sarà il valore numerico del bit e non il valore booleano, quindi "1" e "0".

Esempio:
volendo leggere la variabile <bProva> dichiarata nel DataBase di Movicon.NExT come bit e impostata al valore "1", risulterà:

```
Sub Main Handles .Main
        Debug.Print bProva 'Risultato = -1
        Debug.Print ScriptDocument.GetVariableValue("bProva") 'Risultato = 1
End Sub
```

Variabili senza segno

Il WinWrap non prevede l'utilizzo di variabili senza segno quindi una variabile dell'Address Space di tipo UInt16 (senza segno), UInt32 (senza segno), UInt64 (senza segno), per essere utilizzata nel Basic Script deve essere convertita al fine di non causare un errore di tipo overflow. L'errore di overflow infatti si manifesterebbe qualora la variabile usata nello script superasse il valore di tipo Int16 (ad esempio 32767 per una UInt16) perchè il motore basic non gestisce le variabili di tipo senza segno. **Per evitare questo problema una variabile di tipo UInt32 viene passata al WinWrap sempre come Long.**

Nel caso di variabili contenute all'interno di un folder, data ad esempio la variabile "Var1" all'interno del folder "F1", la notazione diretta da script per riferire tale variabile è:

"F1_Var1"

Sulla base della sintassi dello script, non risulta possibile differenziare le variabili contenute in un folder da quelle che hanno come parte iniziale il nome uguale al folder. Ad esempio "F1\Var1" e "F1_Var1" sarebbe riferite entrambe con la notazione diretta "F1_Var1".
Si deve quindi prestare attenzione a non creare queste sovrapposizioni o nel caso è opportuno utilizzare la notazione indiretta che non si presta a fraintendimenti:
ScriptDocument("F1\Var1") per la variabile all'interno del folder e
ScriptDocument("F1_Var1") per la variabile nella radice.

Variabili Array

Nel caso di utilizzo di variabili di tipo Array è possibile accedere agli elementi specificando il numero dell'elemento tra parentesi tonde utilizzando la sintassi di accesso diretto al tag. Ad esempio supponendo di avere creato nel'eddress space del server il tag "Var_01" di tipo array, si potrà accedere agli elementi del tag con la seguente sintassi:
Var_01(0) = 1
Var_01(1) = 2

L'uso della sintassi diretta è però utilizzabile soltanto nello script lato Client ma non in quello lato Server.
E' tuttavia possibile accedere agli elementi di un Array anche utilizzando le funzioni script con la seguente sintassi:

Script lato Client:
ScriptDocument.GetVariableValue("Var_01")(0) = 1
ScriptDocument.GetVariableValue("Var_01")(1) = 2

Script lato Server:
ServerScriptManager.GetTag("Var_01").Value(0) = 1
ServerScriptManager.GetTag("Var_01").Value(1) = 2

Attenzione però che quando si utilizzano le funzioni script per scrivere un elemento di un Array, anche se questo viene di fatto scritto sul Server, non verrà però notificato al Client, che pertanto visualizzerà il vecchio valore. Perché il Server notifichi il cambiamento al Client è necessario che venga impostata l'intera variabile Array. In questo caso quindi si dovrà utilizzare una variabile di appoggio interna allo script come nel seguente esempio, dove "Folder1\Tag_ArrayFloat" è la variabile Array del Server:

```
'#Language "WWB.NET"
Option Explicit
Sub Main Handles .Main
   Dim myArr() As Single
   Dim nLen As Integer
   Dim iCounter As Integer
   nLen = UBound(Folder1_Tag_ArrayFloat)
   ReDim myArr(nLen)
   myArr = Folder1_Tag_ArrayFloat
   For iCounter = LBound(myArr) To UBound(myArr)
      myArr(iCounter) = myArr(iCounter) + 1
   Next iCounter
   Folder1_Tag_ArrayFloat = myArr
End Sub
```

Variabili Struttura

Nel caso di utilizzo di variabili struttura, quando vengono utilizzate in modo diretto, il carattere di delimitazione fra il nome della variabile e il nome del membro è il punto ("."). Utilizzando invece le funzioni GetVariableValue() e SetVariableValue() il carattere di separazione fra il nome della variabile e il nome del membro sono i due punti (":"). Ad esempio avendo definito la variabile "TagStruct01" di tipo struttura con un prototipo avente i membri "Member_1" e "Member_2", per accedere alla variabile la sintassi sarà:

- Accesso diretto (solo script lato Client): TagStruct01.Member_1 = 1
- Accesso Indiretto (tramite funzioni lato Client): ScriptDocument.SetVariableValue("TagStruct01:Member_1", 1)
- Accesso Indiretto (tramite funzioni lato Server): ServerScriptManager.GetTag("TagStruct01:Member_1").Value = 1

Attenzione! Utilizzando la sintassi diretta è possibile accedere soltanto ai membri semplici definiti nella root del prototipo ma

non ai membri definiti all'interno di folder del prototipo o ai membri definiti di tipo prototipo (struttura annidata). Sarà in questo caso necessario utilizzare le funzioni script per accedere ai membri annidati.

Ad esempio avendo definito la variabile "TagStruct01" di tipo struttura con un prototipo avente il membro "Member_1" all'interno del Folder1 del prototipo, per accedere alla variabile la sintassi sarà:

ScriptDocument.SetVariableValue("TagStruct01:Folder1\Member_1", 1)
ServerScriptManager.GetTag("TagStruct01:Folder1\Member_1").Value = 1

In modo analogo avendo definito la variabile "TagStruct01" di tipo struttura con un prototipo avente il membro "MemberStruct" a sua volta di tipo prototipo e con il membro "Member_1" all'interno del Folder1 del prototipo annidato, per accedere alla variabile la sintassi sarà:

ScriptDocument.SetVariableValue("TagStruct01: MemberStruct \Folder1\Member_1", 1)
ServerScriptManager.GetTag("TagStruct01: MemberStruct \Folder1\Member_1").Value = 1

Variabili all'interno di Folder

Nel caso di utilizzo di variabili contenute all'interno di un folder si dovrà utilizzare il carattere "_" come separatore tra folder e variabile o anche tra folder e folder quando si usa la notazione diretta. Quando invece si utilizzano le funzioni come GetVariableValue() e SetVariableValue() il carattere di separazione diventa il "\". Ad esempio volendo accedere alla variabile "Var1" all'interno del folder "Folder1", la sintassi sarà:

- Accesso diretto:

Folder1_Var1 = 1

- Accesso Indiretto (tramite funzioni)

ScriptDocument.SetVariableValue("Folder1_\Var1", 1)

Oppure con più livelli di folder:

- Accesso diretto:

Folder1_Folder2_Folder3_Var1 = 1

- Accesso Indiretto (tramite funzioni)

ScriptDocument.SetVariableValue("Folder1\Folder2\Folder3\Var1", 1)

Utilizzando l'accesso ai tag in modo diretto bisogna fare attenzione a non avere definito dei nomi di folder e di variabili che possano creare delle omonimie.

Ad esempio se si è definita la variabile "Motor1" all'interno del folder "Machine1" nella root dell'address space, e poi si è anche definita una seconda variabile direttamente nella root dell'address space con il nome "Machine1_ Motor1", l'intellisense dello script non sarà in grado di distinguere le due variabili perché la sintassi utilizzata per accedere ai due tag risulterà uguale, ovvero "Machine1_ Motor1".

In casi come questo sarà necessario ricorrere all'uso delle funzioni GetVariableValue() e SetVariableValue() per distinguere i due tag.

TimeStamp e Qualità

E' possibile, da script, leggere la sua qualità o il TimeStamp di una variabile tramite la notazione diretta:
"NomeVar_Quality", "NomeVar_TimeStamp".
Non è quindi buona norma creare variabili che abbiano il nome di una variabile esistente ed un suffisso del tipo:
"_quality", "_timestamp" in quanto con la notazione diretta risulterebbe impossibile distinguere se ci si vuole riferire alla qualità della variabile o alla variabile che ha quel nome.
Si veda anche l'uso delle funzioni "GetVariableQuality", "GetVariableTimeStamp" che possono essere usate per riferire la qualità ed il timestamp della variabile in modo univoco anche nella situazione sopra descritta.

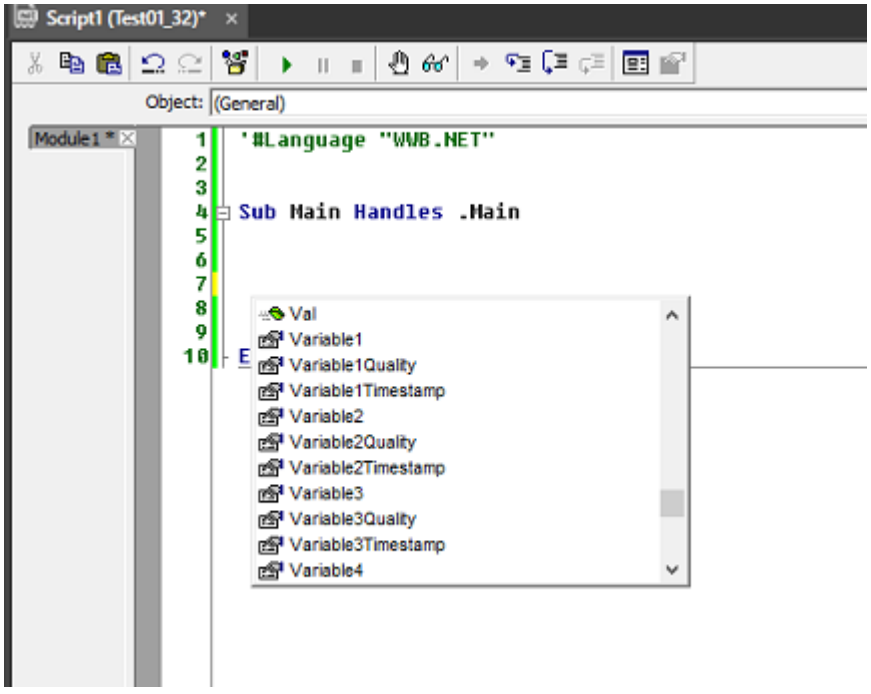
Variabili Locali e di Sistema del Client

Da codice script è possibile anche accedere ai Tag Locali e ai tag di Sistema del Client. Per i tag locali digitare "TemporaryVariables" seguito da un punto e apparirà la lista dei membri che si possono selezionare. Per i tag di sistema digitare "SystemVariables" seguito da un punto e apparirà la lista dei membri che si possono selezionare. Ad esempio:

- TemporaryVariables.LocalTag1
- SystemVariables.CurrentUser

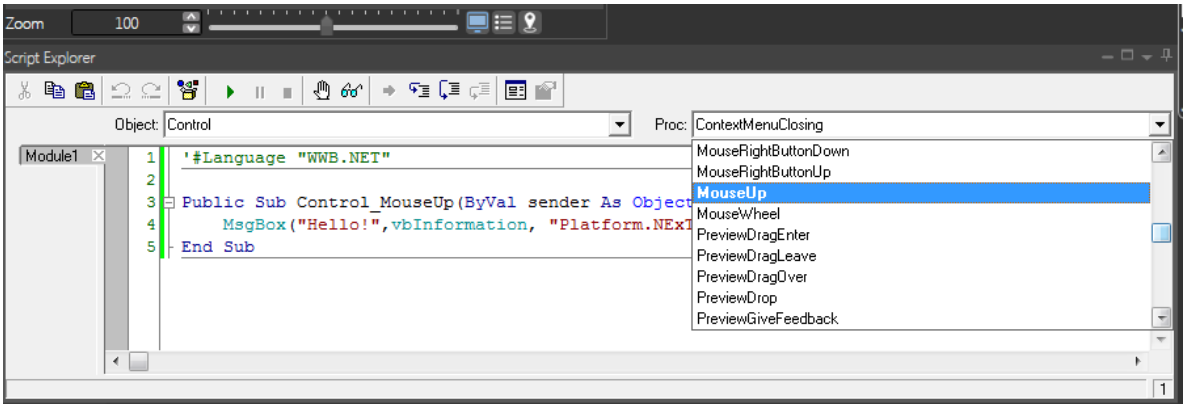
Intellisense nello Script

Le risorse Basic Script sono in grado di identificare le variabili del progetto quando queste sono scritte in modo diretto, e nel caso il nome del tag viene scritto di colore blu. Risulta piuttosto comodo utilizzare il comando "CTRL+Barra Spaziatrice" per avere l'elenco dei metodi e proprietà disponibili ma anche dei tag del progetto. Questo consente ad esempio di poter "selezionare" il nome di un tag e non doverlo scriverlo per intero. Premendo i tasti "CTRL+Barra Spaziatrice" all'interno dello script apparirà quindi una list box, come quella mostrata in figura, che potrà essere scrollata per selezionare il Tag desiderato:



2.3.9. Basic Script WWB.NET™ nel Codice degli Oggetti

I simboli all'interno dei sinottici e i sinottici possono contenere del codice script. La gestione di questo codice basic inserito negli oggetti è differente da quella delle risorse Basic Script, la cui esecuzione è comandata nelle logiche del progetto. All'interno dell'oggetto sono resi disponibili una serie di eventi che se selezionati e quindi inseriti all'interno dell'editor basic script consentono di inserirvi del codice. Con il progetto in esecuzione in concomitanza della generazione di questi eventi nel progetto verranno eseguite le funzioni basic programmate al loro interno.
In base all'oggetto in cui si edita del codice è possibile avere a disposizione degli eventi oltre che delle proprietà e dei metodi diversi. Per una spiegazione dettagliata di ciascuno di questi si rimanda all'elenco presente nei paragrafi specifici.



Importante è ricordare che il codice associato ad un evento di un oggetto è eseguito solamente se l'oggetto è caricato in memoria e quindi gestito da Movicon.Next. Per esempio un disegno a cui viene associato del codice è eseguito solamente quando il sinottico contenitore è caricato.

Va considerato che al caricamento della pagina sinottico il codice all'interno dei disegni non viene subito inizializzato ma soltanto quando è necessario. Se un simbolo contiene l'evento "Loaded" allora Movicon.Next è costretto ad inizializzare da subito il codice basic script contenuto in quel disegno. Questo significa che il caricamento della pagina risulterà più veloce se i disegni ad essa associata non contengono l'evento "Loded". Naturalmente questo non significa che l'evento "Loded" non debba essere utilizzato, ma conviene gestirlo solo se necessario.
Le variabili del progetto (Address Space di Platform.NExT) possono essere utilizzate direttamente con il loro nome oppure utilizzando le apposite funzioni della "Document.GetVariableValue()", "Document .SetVariableValue()" per il codice editato a livello degli oggetti di un sinottico.

Il codice script dei disegni viene caricato solo quando si rende necessario il suo utilizzo e non quindi al caricamento della pagina.

2.3.10. Script Lato Server

Il codice avviato nel contesto del Server viene ovviamente eseguito sul pc nel quale è avviato il Server (indipendentemente quindi dal fatto che sia avviato anche il Client). Per

editare il codice script lato Server è necessario inserire il codice all'interno dei Tag, come eventi generati dal sistema o come metodi personalizzati. Il codice lato Server viene caricato non appena viene istanziata la variabile sulla quale è definito lo script.

Se si imposta un breakpoint nel codice del Tag, all'avvio del Server, anche se il metodo non è stato ancora invocato, si aprirà la finestra di debug del winwrap.

Metodi di una Variabile lato server

In Movicon.Next E' possibile editare dei metodi per una variabile. L'editazione di tali metodi viene effettuata selezionando la variabile nell'Address Space e aprendo la finestra "Script Explorer".
Si possono creare delle sub-routine con parametri d'ingresso.
Tale metodo è anche richiamabile dal client, dopo avere avviato la parte server del progetto, nel quale è definito il metodo, trascinando ad esempio il metodo su di un pulsante della pagina client.
lo stesso risultato si potrà ottenere aggiungendo un comando di tipo "Call Method" al pulsante del client e selezionando nella scheda "OPC UA Browser" il metodo da richiamare.
Gli eventuali parametri, verranno visualizzati in una finestra a seguito del trascinamento del metodo della variabile dalla finestra del server sul pulsante al quale verrà applicato il comando "Call Method".
In alternativa se si usa la scheda "OPC UA Browser" con il pulsante "Data Settings" si potrà impostare il valore del parametro.
Durante l'operazione d'associazione del metodo al pulsante, nel caso sia presente il parametro d'ingresso, verrà anche richiesto di impostare un valore da associare ad esso.
Tali routine vengono eseguite nel contesto del server.

Attenzione! E' richiesta comunque una sintassi che imponga il prefisso "Method" al nome delle routine create.

La dimensione del codice script che può essere inserito all'interno di un singolo Tag non può superare i 32767 caratteri.

Per leggere e scrivere il valore della variabile stessa all'interno del metodo si può usare la proprietà "ServerScriptManager.Value", mentre per accedere ad 'un'altra variabile si usa la funzione ServerScriptManager.GetTag("NomeVar").value.

Le variabili possono poi essere definite di tipo metodo (come tipo di dato), in relazione ad un driver che espone dei metodi invocabili tramite la variabile.
Es:

```
Public Sub Method_Test()'ByVal par1 As String
    Try
        Dim path As String = "c:\temp\MyTest.txt"
        Dim fi As FileInfo = New FileInfo(path)

        If fi.Exists() Then
            fi.Delete()
        End If
```

l'istruzione seguente scrive un messaggio nel file di Log:

```
        debug.print "Scrivo del testo nel file...."
        'Create the file.
        Dim fs As FileStream = fi.Create()
        Dim info As Byte()
        info= New UTF8Encoding(True).GetBytes("Scrivo del testo nel
file....")
        'Add some information to the file.
        fs.Write(info, 0, info.Length)
        fs.Close()
        'Open the stream and read it back.
        Dim sr As StreamReader = fi.OpenText()
        sr.Close()
    Catch ex As System.Exception
        Debug.Print "Errore metodo Test della variabile VarBoolean: " &
ex.Message
    Finally
    End Try
End Sub
```

Eventi di una variabile lato server

All'interno delle variabili possiamo anche trovare alcuni eventi utilizzabili dall'utente finale: ServerScriptManager espone gli eventi Init, Terminate, DoEvents, mentre AnalogItemState l'evento StateChanged. Per maggiori info vedi il capitolo relativo alle API Basic Script.

Uso di variabili Array

All'interno dello script del server è possibile anche accedere agli elementi di una variabile array. La sintassi è la seguente:
ServerScriptManager.GetTag("Tag_Array1").Value(0)
Attenzione però che quando si scrive un singolo elemento di un array il Server non notifica tale cambiamento agli eventuali Client connessi. Perché la notifica avvenga correttamente è necessario che venga aggiornata tutta la variabile array. La procedura corretta in questi casi quindi è quella di copiare la variabile array su una variabile di appoggio dello script, modificare l'elemento della variabile di appoggio e poi ricopiare la variabile di appoggio sulla variabile array del server. Ad esempio:

```
Public Sub AnalogItemState_StateChanged(ByVal context As Opc.Ua.ISystemContext,
ByVal node As Opc.Ua.NodeState, ByVal changes As Opc.Ua.NodeStateChangeMasks)
Handles AnalogItemState.StateChanged
    Dim appArray() As Single
    appArray() = ServerScriptManager.GetTag("Tag_Array1").Value
    appArray(0) = 99
    ServerScriptManager.GetTag("Tag_Array1").Value = appArray()
End Sub
```

INDUSTRIAL AUTOMATION SOFTWARE

progea

Progea Srl
Via D'Annunzio, 295
I-41123 Modena
info@progea.com
Tel +39 059 451060

Progea International SA
via Sottobisio, 28
6828 Balerna (CH)
international@progea.com
Tel +41 91 96 76 610

Progea Deutschland GmbH
Marie-Curie Str., 12
D-78048 VS Villingen
info@progea.de
Tel +49 (0)7721 99838 0

Progea North America Corp.
2380 State Road 44, Suite C
Oshkosh, WI 54904
info@progea.us
Tel. +1 (888) 305-2999