Sommario

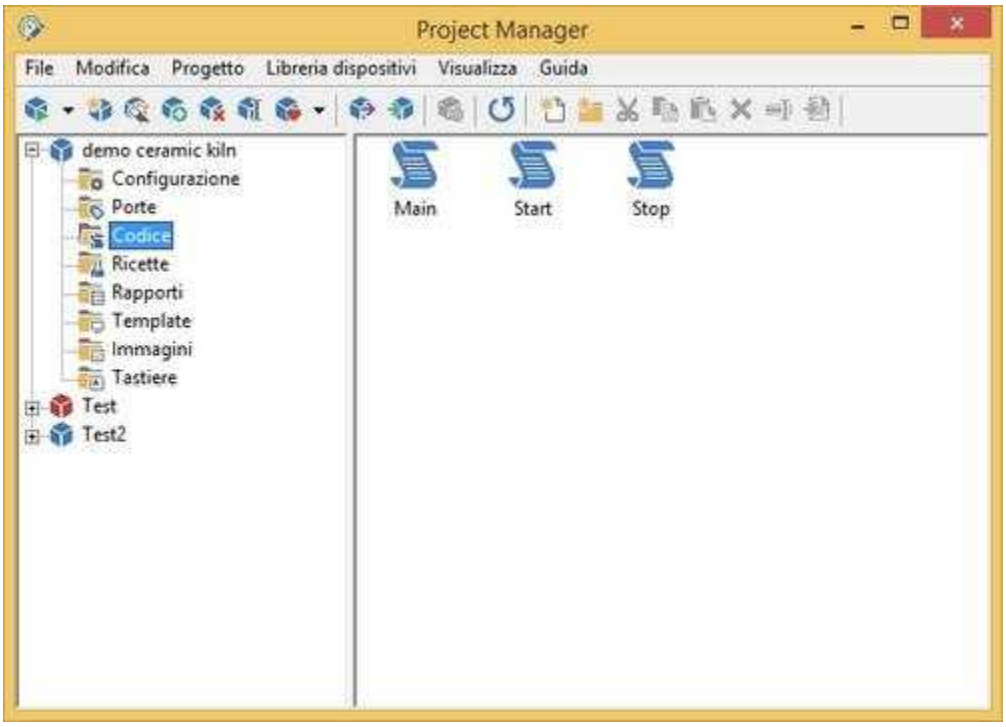
1 Introduzione

Spesso, in un progetto di supervisione, nasce l'esigenza di eseguire particolari operazioni al verificarsi di alcune condizioni o su richiesta dell'operatore. Alcuni esempi possono essere l'eseguire un calcolo di una formula complessa, inviare mail o SMS, leggere o scrivere file di testo, operare direttamente sui valori delle porte, manipolare alcuni oggetti del sinottico visualizzato in quel momento, etc.
Code Builder è un editor progettato per facilitare la scrittura di tali funzioni. Permette di scrivere pagine di codice e di controllarne la sintassi; mette a disposizione una serie di strumenti di ricerca e manipolazione del testo tipiche degli editor; inoltre, per semplificare e ridurre la presenza degli errori, le sequenze di caratteri sono evidenziate con una differente colorazione in relazione al ruolo che ricoprono nella sintassi.

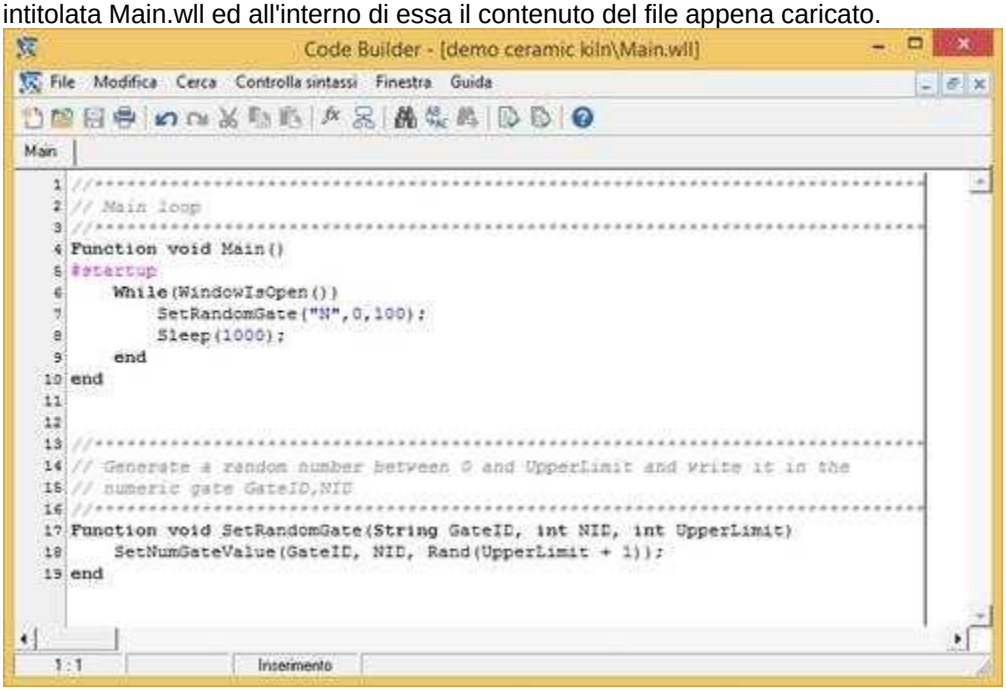
2 Il primo passo

La creazione di un nuovo file di codice vuoto è effettuata nel *Project Manager*, selezionando la cartella *Codice* e successivamente spostando il mouse sulla finestra di destra e premendo il tasto destro: comparirà quindi la voce di menu "*Nuovo* | *File di codice"*

Code Builder viene avviato automaticamente da *Project Manager* attraverso il doppio click sull'icona relativa ad una qualsiasi pagina di linguaggio precedentemente creata nel *Project Manager* stesso.



Qui di seguito si vede *Code Builder* aperto e nello spazio di lavoro si può osservare una finestra intitolata Main.wll ed all'interno di essa il contenuto del file appena caricato.



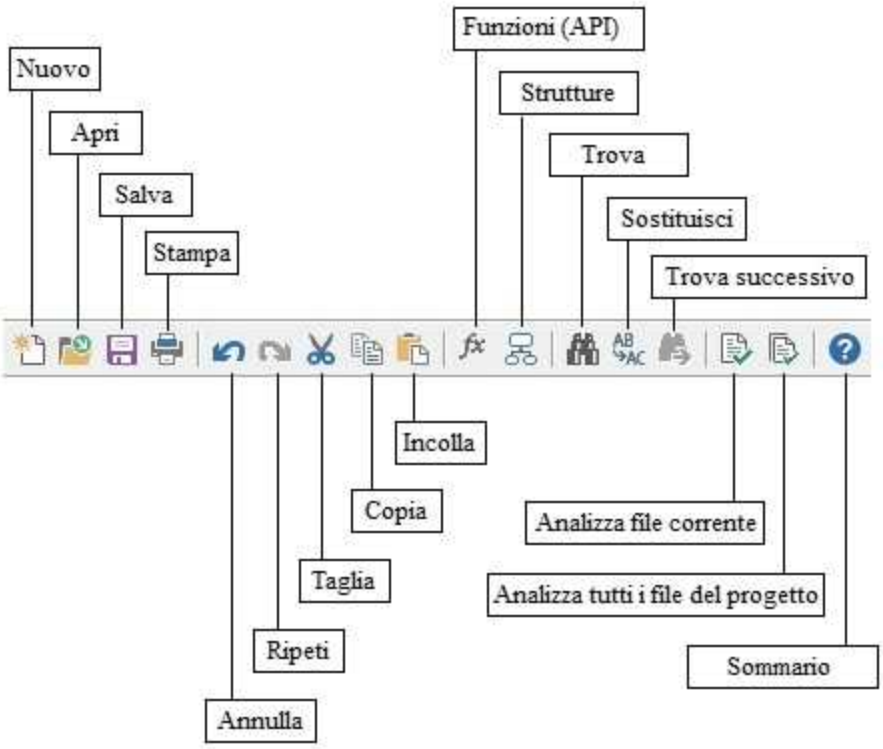
L'ambiente integrato si presenta composto da menù, pulsanti di scelta rapida, da una barra di stato per le informazioni (in basso) e da uno spazio di lavoro vuoto.
Le finestre di codice su cui si lavora saranno visualizzate all'interno dello spazio di lavoro.

Si può notare, all'interno della nuova finestra, la differente colorazione che assume il codice dell'esempio. Il testo è colorato in base all'analisi del riconoscitore interno di *Code Builder* che categorizza gli elementi del testo. Ovviamente gli attributi di ogni categoria possono essere modificati

a proprio piacimento, mentre, le regole che distinguono le classi di elementi, sono fisse e discendono direttamente dalla sintassi del linguaggio.
Si può notare, sulla barra di stato, degli elementi che prima non erano visibili; sono rispettivamente la posizione del cursore (riga : colonna) nel primo rettangolo e la parola *inserimento* nel terzo rettangolo; quest'ultima indica che ogni carattere che verrà permuto sulla tastiera verrà inserito alla posizione attuale del cursore. È anche possibile decidere che i caratteri, invece di essere inseriti, vengano rimpiazzati. Premendo il tasto INS è possibile notare due cambiamenti: la forma del cursore che diventa più larga e la comparsa della parola *sovrascrittura* sulla barra. In questo modo i caratteri non sono più inseriti nel testo ma lo sovrascrivono; per tornare alla modalità d'inserimento si può semplicemente premere di nuovo INS.

3 La speedbar

La speedbar mette a disposizione dei pulsanti di scelta rapida per funzioni frequentemente richiamate dai menu.
Posizionando il puntatore del mouse su uno di questi pulsanti si potrà osservare la comparsa di una breve spiegazione della funzione svolta.



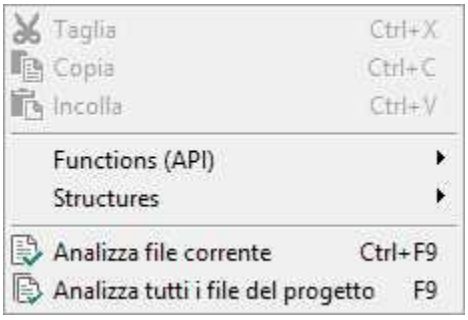
4 Utilizzo della tastiera

Tramite la tastiera è possibile accedere a diverse funzioni messe a disposizione dai menu; si è visto in precedenza che accanto ad alcune voci di menu è rappresentata una combinazione di tasti.

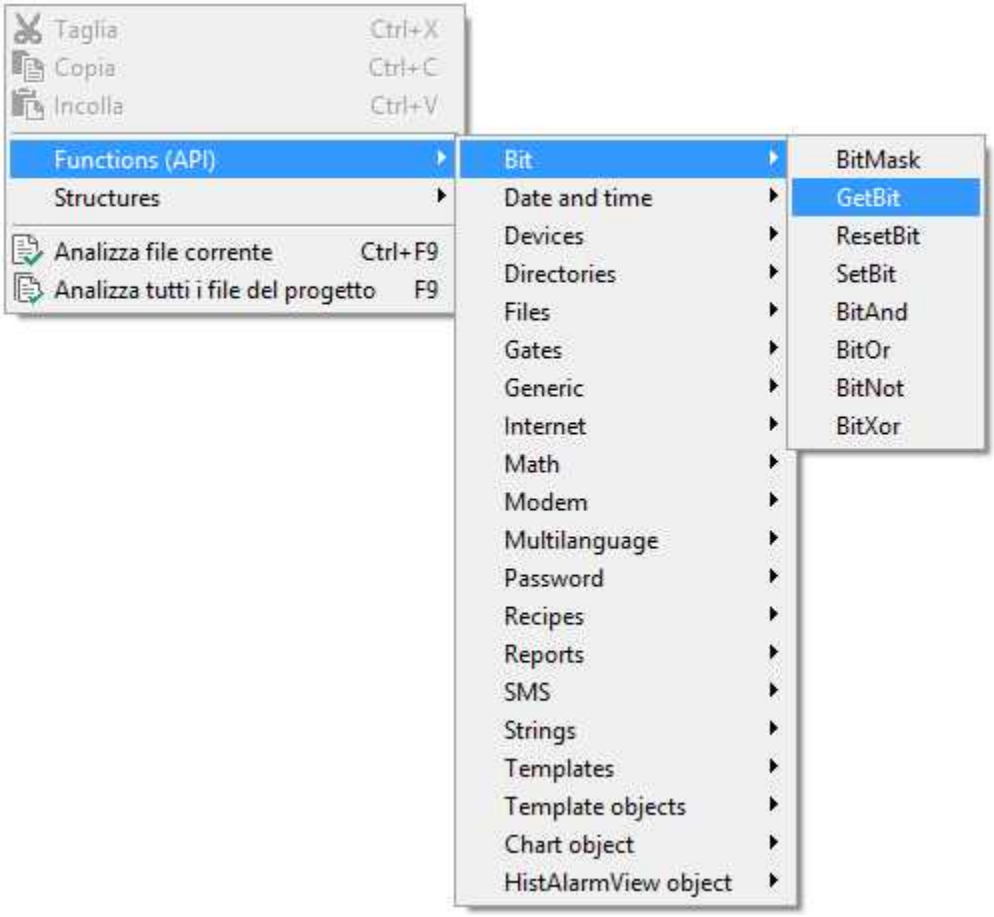
Tasti	Funzione	Dispobile con pressione tasti	Disponibile nel menù
CTRL + N	crea una nuova finestra di codice	✓	✓
CTRL + O	apre un file di codice di *WinLog*	✓	✓
CTRL + S	salva il file su cui si sta lavorando	✓	✓
CTRL + Z	ripristina l'ultima cancellatura fatta	✓	✓
CTRL + C	copia il testo selezionato nella clipboard	✓	✓
CTRL + X	elimina il testo selezionato	✓	✓
CTRL + V	copia il testo della clipboard alla posizione del cursore	✓	✓
CTRL + F	ricerca del testo	✓	✓
CTRL + R	cerca e rimpiazza del testo	✓	✓
CTRL + F9	controlla la sintassi della finestra di codice attuale	✓	✓
F3	ripete l'ultima ricerca	✓	✓
F9	controlla la sintassi di tutto il progetto	✓	✓
F1	richiama l'aiuto contestuale	✓	✓
INS	modifica lo stato d'inserimento	✓	✗
CTRL+ Y	cancella la riga su cui è posizionato il cursore	✓	✗
ALT + N	porta in primo piano la finestra di codice successiva	✓	✗
ALT + P	porta in primo piano la finestra di codice precedente	✓	✗
CTRL+ LEFT	sposta il cursore di una parola a sinistra	✓	✗
CTRL +RIGHT	sposta il cursore di una parola a destra	✓	✗
CTRL + HOME	sposta il cursore sulla prima riga del codice	✓	✗
CTRL + END	sposta il cursore sull'ultima riga del codice	✓	✗

5 Utilizzo del mouse

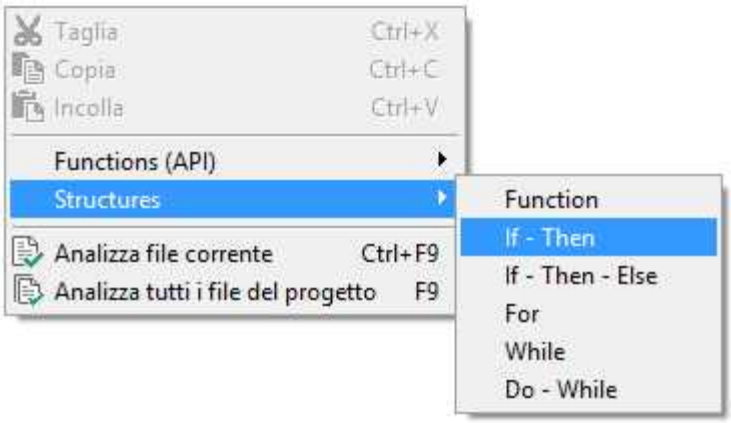
Il pulsante destro del mouse offre la scorciatoia per richiamare alcune funzioni dell'ambiente ed alcune funzioni che facilitano la creazione del codice.



API mette a disposizione una serire di sottomenu con tutte le funzioni delle librerie WL



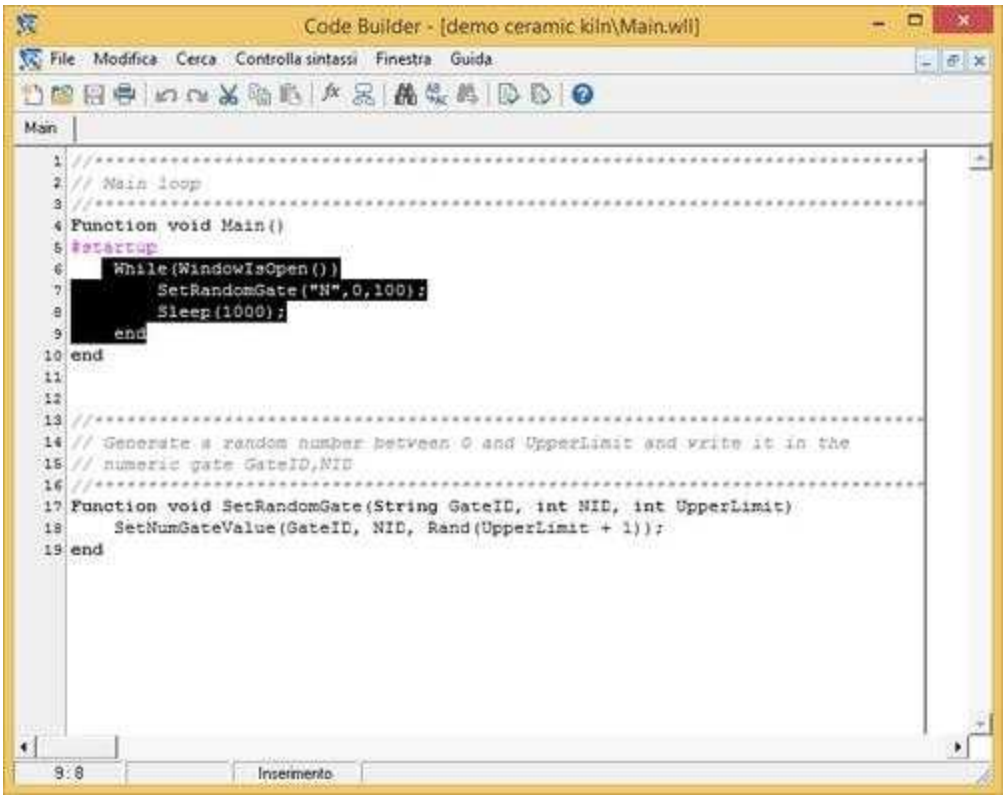
Strutture linguaggio offre, a scelta, in un menu, le strutture comunemente utilizzate durante la stesura di codice.



6 Utilizzo della clipboard

La clipboard è un'area, non accessibile direttamente, in cui si può inserire oppure da cui si può prelevare del testo.

Il testo su cui si vuole effettuare delle operazioni di trasferimento o cancellazione deve per prima cosa essere selezionato. Per selezionare il testo è necessario tenere premuto il tasto SHIFT e muovere il cursore nello schermo in maniera da evidenziare il testo desiderato, oppure con il mouse tenendo premuto il tasto sinistro.



Tutto il testo evidenziato sarà quello su cui si effettuerà l'operazione desiderata scelta tra quelle del menu modifica.
Dopo aver selezionato il testo desiderato è possibile:

Copiarlo nella clipboard
Tagliarlo ovvero eliminarlo dal codice e spostarlo nella clipboard

Per *incollare* del testo dalla clipboard non è necessario effettuare nessuna selezione, il testo contenuto nella clipboard sarà inserito alla posizione del cursore.

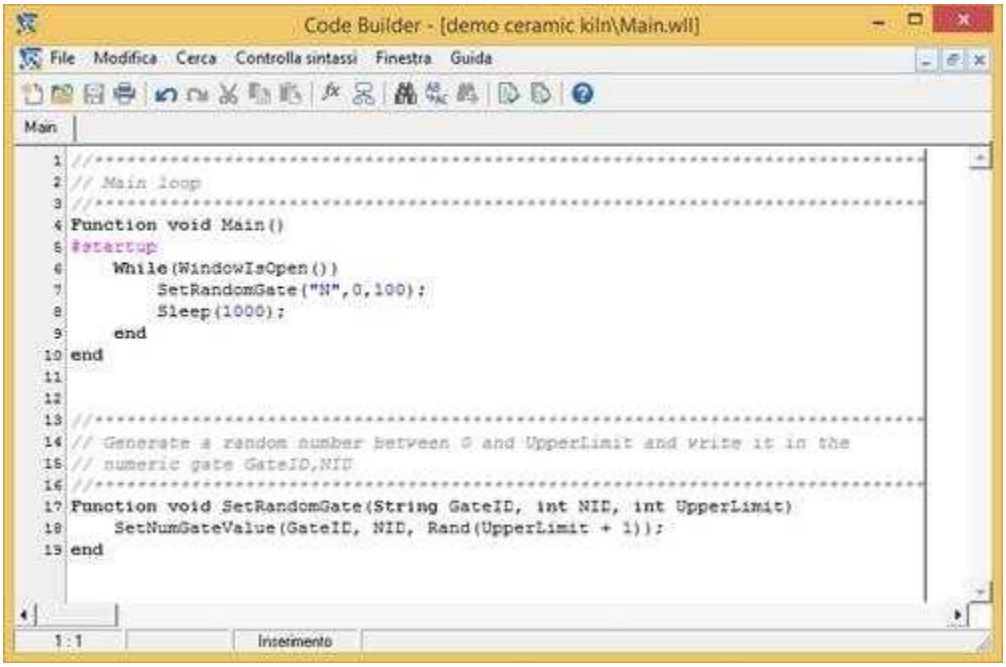
Il pulsante sulla speedbar che denota la funzione *incolla* è colorato soltanto quando la clipboard contiene qualcosa da poter copiare.
In modo del tutto analogo, i pulsanti per la copia e l'eliminazione di blocchi di testo, sono evidenziati solamente quando, all'interno del codice cui si sta lavorando, esiste del testo selezionato.

7 I menu

7.1 Introduzione

Ora si tratteranno dettagliatamente le funzioni che i menu mettono a disposizione.

Ci sono due modi per accedere ai menu mentre si sta lavorando, il primo prevede l'utilizzo del mouse ed il secondo invece quello del tasto ALT in combinazione con la lettera del menu sottolineata.

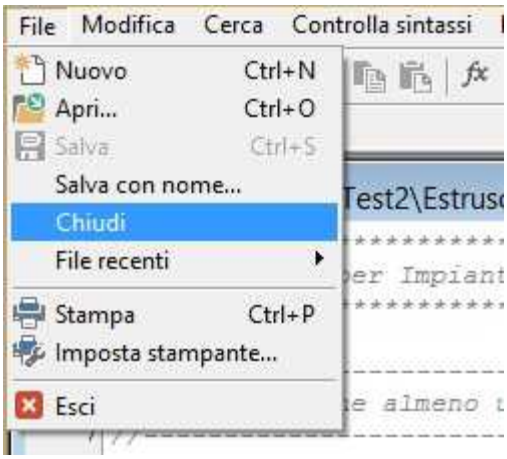


Le funzioni messe a disposizione dai menu sono applicate alla finestra di codice corrente, quella che rispetto alle altre è in primo piano

7.2 File

7.2.1 A cosa serve

Il menu *File* contiene tutte quelle funzioni necessarie alla memorizzazione permanente e alla stampa del codice.



Da questo menu si possono aprire file già esistenti, salvare quello su cui si sta lavorando oppure stamparlo o chiuderlo. Accanto ad ogni scelta del menu si può notare la specifica di uno o più tasti; ad esempio alla scelta *nuovo* corrisponde la combinazione CTRL+N.
I tre punti di sospensione che compaiono in una scelta indicano che questa apre una finestra di

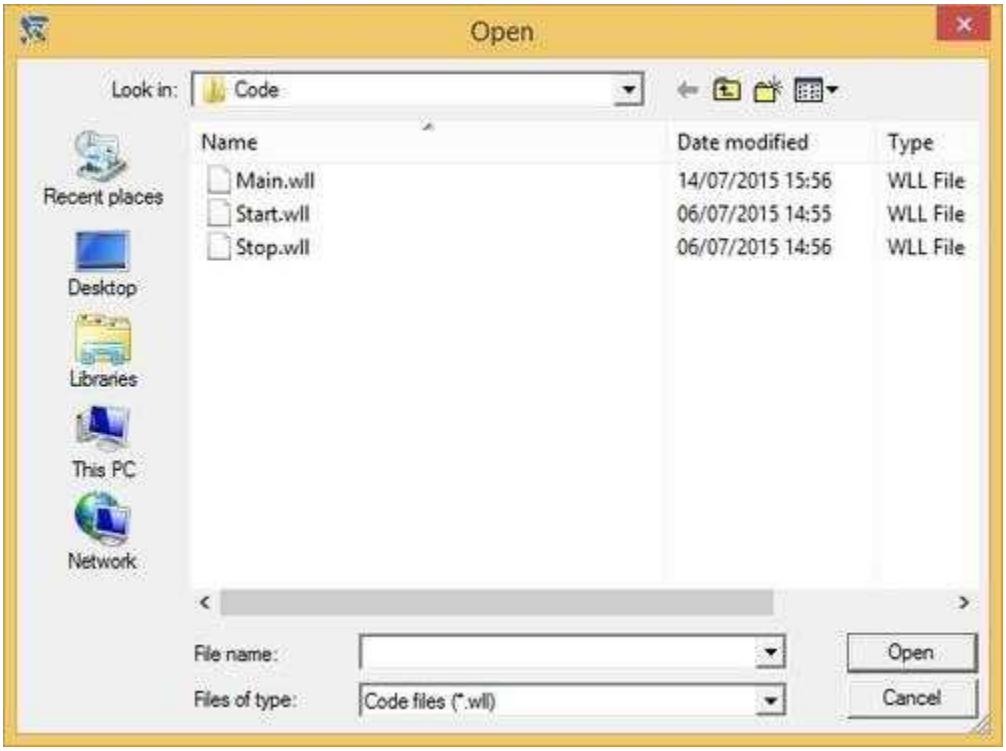
dialogo.

7.2.2 Nuovo

Nuovo: apre una finestra vuota nello spazio di lavoro.

7.2.3 Apri

Apri: mostra una finestra di dialogo in cui è possibile selezionare il file su cui si vuole lavorare.



Una volta selezionato il file desiderato basta premere il pulsante *Apri*.

7.2.4 Salva

Salva: memorizza il file cui si lavora; la prima volta che si salva un file, creato col comando *nuovo,* appare una finestra di dialogo simile alla precedente, in cui è possibile salvare il file col nome creato automaticamente oppure sceglierne uno nuovo. I dati, invece, saranno salvati senza passare per la finestra di dialogo se il file è già stato salvato almeno una volta.

7.2.5 Salva con nome

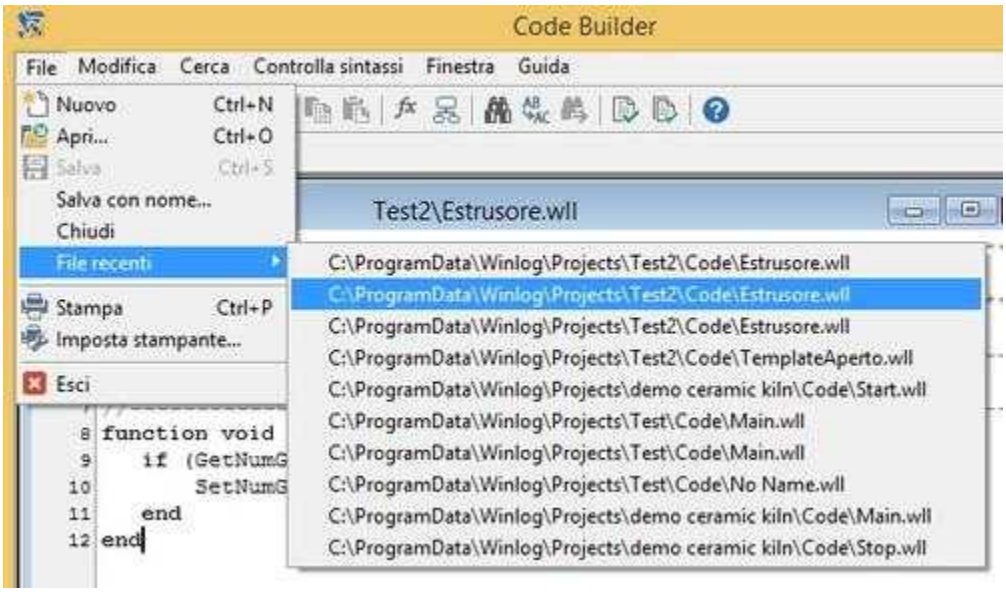
Salva con nome: Memorizza i dati come per la funzione salva l'unica differenza sta nel fatto che apre ogni volta la finestra di dialogo per il salvataggio del file

7.2.6 Chiudi

Chiudi: chiude la finestra di codice attualmente in primo piano.

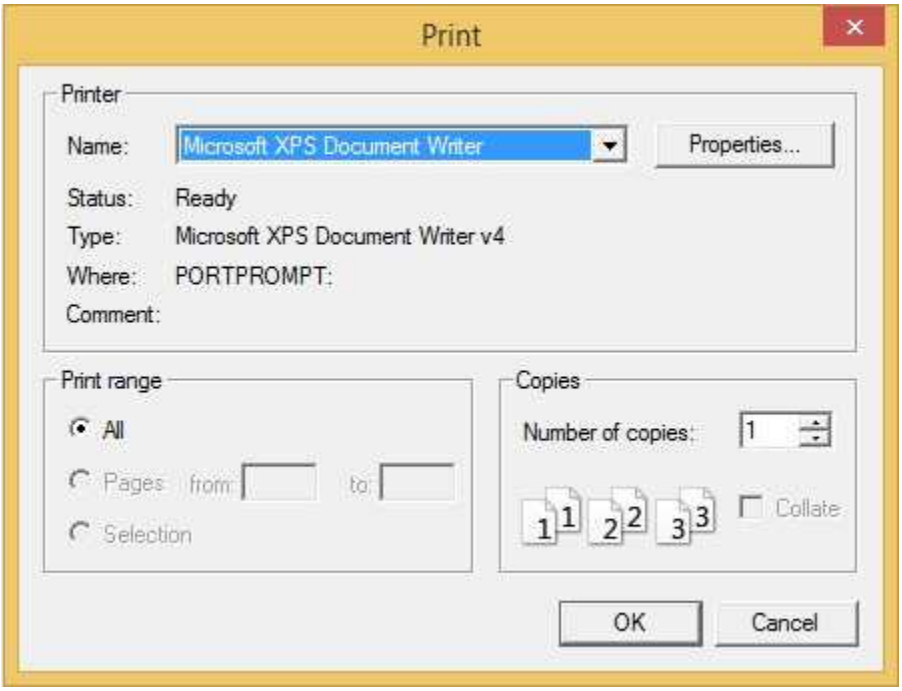
7.2.7 File recenti

File recenti:mantiene una lista dei 10 file più recenti.



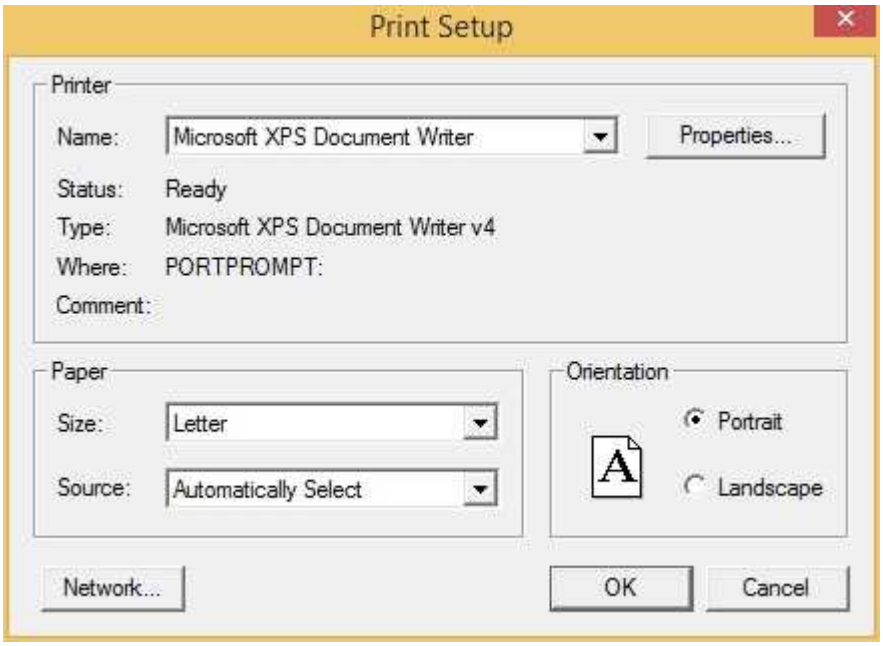
7.2.8 Stampa

Stampa: invia il contenuto della finestra di codice su cui si lavora alla stampante



7.2.9 Imposta stampante

Imposta stampante: configura le opzioni di stampa



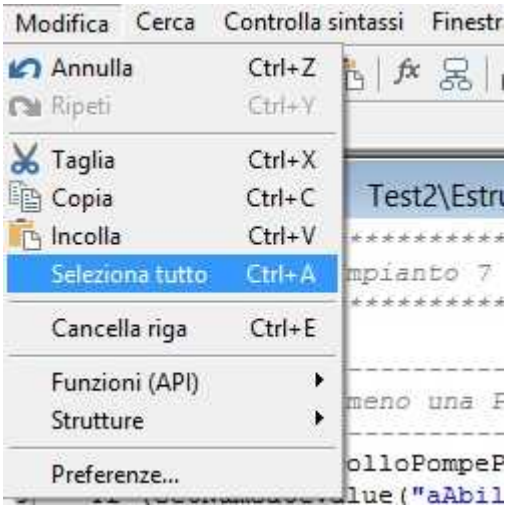
7.2.10 Esci

Esci: Termina la sessione di lavoro con CBuilder

7.3 Modifica

7.3.1 A cosa serve

Il menu *modifica* contiene le funzioni necessarie alla manipolazione dei blocchi di testo, alla correzione di operazioni errate ed alla configurazione dell'ambiente di lavoro.



7.3.2 Annulla

Annulla: si limita a recuperare l'ultimo testo cancellato, che può essere un carattere oppure un blocco di caratteri.

7.3.3 Ripeti

Ripeti: ripristina l'ultima operazione di 'Annulla' fatta

7.3.4 Taglia

Taglia: elimina il testo selezionato nella finestra di codice e lo copia in un'area di memoria (clipboard) dalla quale può essere letto in seguito.

7.3.5 Copia

Copia: prende il blocco di caratteri selezionato e lo copia nella clipboard .

7.3.6 Incolla

Incolla: legge il contenuto della clipboard e lo inserisce nel codice alla posizione attuale del cursore.

7.3.7 Seleziona tutto

Seleziona tutto: permette di selezionare tutto il testo del file corrente

7.3.8 Cancella linea

Cancella linea: permette di cancellare la linea di codice in cui è posizionato il cursore, senza salvarlo nella clipboard

7.3.9 Funzioni (API)

Funzioni (API): permette di scegliere e inserire nel codice una funzione, scelta dalla lista di API disponibili

7.3.10 Strutture

Strutture: permette di scegliere e inserire nel codice una struttura, scelta dalla lista di strutture disponibili

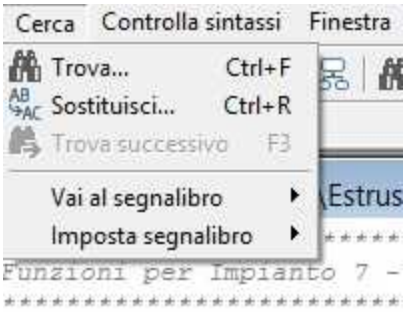
7.3.11 Preferenze

Preferenze: apre una finestra di dialogo dove è possibile configurare l'ambiente a proprio piacimento. La configurazione dell'ambiente integrato è discussa in un'altra sezione.

7.4 Cerca

7.4.1 A cosa serve

In questo menu sono contenute le funzioni per la ricerca e la sostituzione di testo.



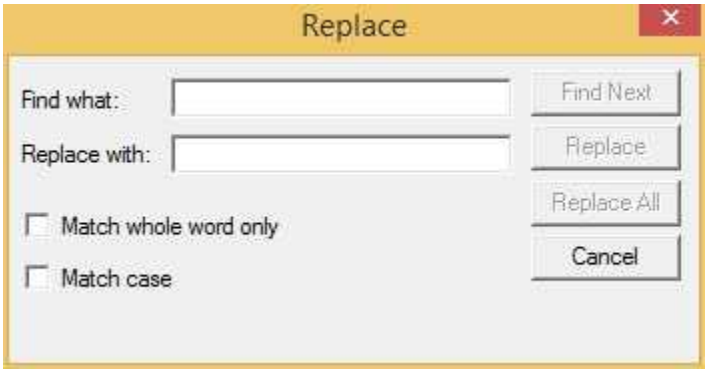
7.4.2 Trova

Trova: apre una finestra di dialogo in cui si può inserire una o più parole da cercare.

All'interno di questa finestra si possono specificare i parametri della ricerca. Dato che l'operazione è compiuta dalla posizione corrente del cursore, si può decidere di dirigersi verso l'inizio (*Su*) oppure verso la fine (*Giù*) del codice. Si può richiedere che la parola sia cercata come è stata inserita, marcando il riquadro *Maiuscole/Minuscole*, in questo modo la ricerca distinguerà tra lettere maiuscole e minuscole. Dopo avere inserito il testo nel riquadro, premendo il pulsante *successivo*, il cursore sarà posto sulla prima occorrenza del testo trovata

7.4.3 Sostituisci

Sostituisci: apre una finestra di dialogo (figura .11) in cui si specifica del testo da cercare e sostituire con uno nuovo.



È possibile specificare se la ricerca deve distinguere tra minuscole e maiuscole come per la funzione precedente. Il pulsante *sost. tutto* ripete la sostituzione inserita per tutte le occorrenze trovate nel test

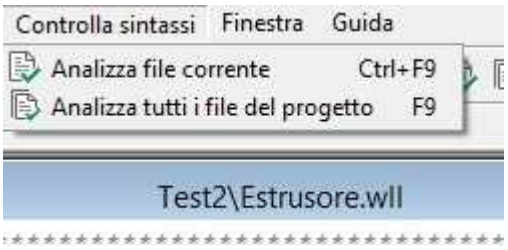
7.4.4 Trova successivo

Trova successivo: ripete l'ultima ricerca eseguita con la funzione, senza aprire nessuna finestra di dialogo.

7.5 Controlla sintassi

7.5.1 A cosa serve

Questo menu mette a disposizioni le funzioni necessarie per il controllo sintattico del codice.



7.5.2 Analizza file corrente

Analizza file corrente: esamina il file su cui si sta lavorando alla ricerca di eventuali errori sintattici.

7.5.3 Analizza tutti i file del progetto

Analizza tutti i file del progetto: esamina tutto il progetto, nel suo complesso, alla ricerca di errori. In caso di errore, il cursore sarà posizionato dove è stato rilevato l'errore.

7.6 Finestra

7.6.1 A cosa serve

All'interno del menu risiedono i comandi per la gestione delle finestre nello spazio di lavoro.
Nella parte bassa del menu è possibile vedere la lista delle finestre di codice presenti: per accedere ad una di queste è sufficiente selezionarla dal menu.

7.6.2 Cascata

Cascata: ridimensiona e posiziona tutte le finestre nello spazio di lavoro, sovrapponendole l'una all'altra ma spostandole leggermente in modo da poter accedere ad ognuna.

7.6.3 Affianca

Affianca: divide lo spazio di lavoro in maniera equa tra tutte le finestre di codice.

7.6.4 Arrangia icone

Arrangia icone: ordina le icone delle finestre minimizzate.

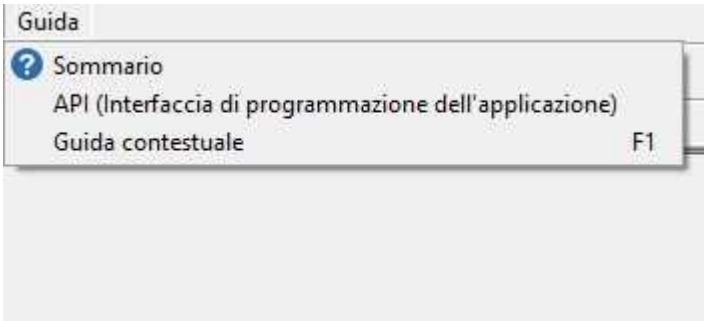
7.6.5 Minimizza tutto

Minimizza tutto: riduce tutte le finestre di codice alle minime dimensioni e le posiziona ordinatamente.

7.7 Guida

7.7.1 A cosa serve

In caso di necessità può essere utile consultare la guida oppure l'aiuto contestuale, accessibili entrambi nel menu *Aiuto.*



7.7.2 Sommario

Sommario: mostra gli argomenti contenuti nell'help.

7.7.3 API (Interfaccia di Programmazione dell'Applicazione)

API (Interfaccia di Programmazione dell'Applicazione): mostra la guida relativa alle API.

7.7.4 Guida contestuale

Guida contestuale: mostra l'indice degli argomenti contenuti nell'help con la possibilità della ricerca per parola chiave.

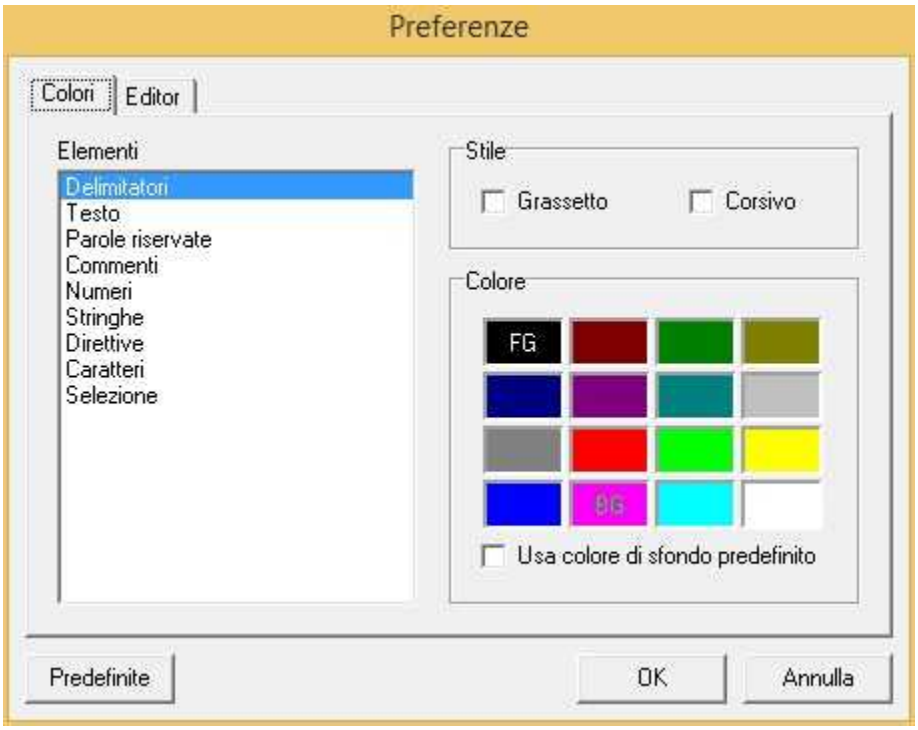
8 Configurare l'ambiente integrato

8.1 Introduzione

Ci sono aspetti dell'ambiente integrato che possono essere configurati. Questo è possibile scegliendo *Preferenze* nel menu *Modifica*; apparirà un modulo con diverse pagine: ogni pagina ha un nome differente, colori *, editor* , ecc.
Per accedere ad una pagina differente da quell'attuale è necessario cliccare sul corrispettivo nome. Si vedrà in dettaglio cosa è possibile configurare in questo modulo.

8.2 Colori

È possibile modificare la colorazione e lo stile degli elementi del testo a proprio piacimento. Le regole che distinguono gli elementi del testo provengono direttamente dalla sintassi del linguaggio.
Selezionando un elemento della lista *Elementi*, la pagina si aggiornerà automaticamente, mostrando la configurazione per quell'elemento.



Ad esempio, come mostrato in figura, l'elemento *parole riservate* ha come stile il *grassetto*, come colore di sfondo il bianco e come colore del testo il nero.

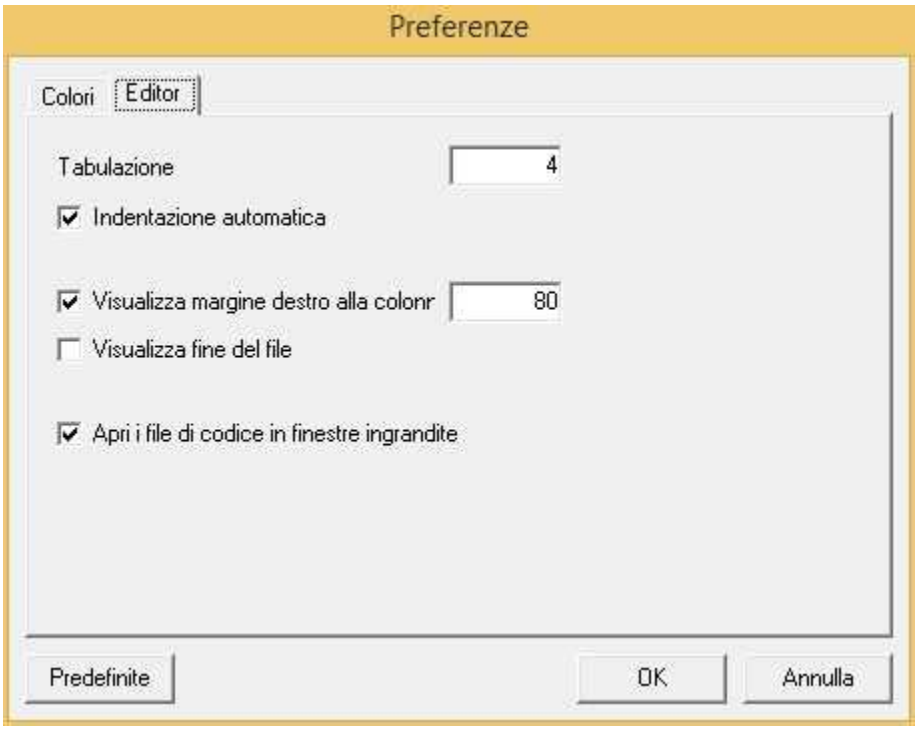
```
//---------------------------------------
// Creazione nuova Ricetta di Produzione
//---------------------------------------
Function Void Nuova_Ricetta()
#Macro
   String name = InputDialog("Nome della Ricetta","Crea Nuova Ricetta","");
   if (name!="") then
      if (RecipeCreate("Ricette di Produzione",name,true)) then
         RecipeImport("Ricette di Produzione",name,true);
      end
   end
End

//-----------------------------------------
// Carica i valori delle porte della discesa
//-----------------------------------------
Function Void CaricaValoriPorte(String prefix, Int num)
  gate1 = GetNumGateValue(prefix+"P.Sp",num);  // Rulli Inferiori - Velocità di SetPoint
  gate2 = GetNumGateValue(prefix+"S.Sp",num);  // Rulli Inferiori - Tensione Filo
  gate3 = GetNumGateValue(prefix+"Man.O",num); // Rulli Inferiori - Uscita in Manuale
  gate4 = GetNumGateValue(prefix+"Mode",num);  // Rulli Inferiori - Modo di Funzionamento
```

Per cambiare lo stile di un elemento è sufficiente marcare o cancellare una delle caselle raggruppate sotto la voce *stili*, analogamente, per scegliere il colore, sarà necessario cliccare su uno dei quadrati colorati: cliccando col tasto sinistro del mouse si seleziona il colore del testo (FG) mentre, utilizzando il destro, si definisce il colore dello sfondo (BG).
È possibile configurare ogni elemento secondo il proprio gusto e confermare le modifiche effettuate premendo il pulsante *OK*; se si desidera annullare è sufficiente premere il pulsante *Annulla*.
Per ripristinare le opzioni predefinite si può premere il pulsante *Predefinite.*
La voce "*utilizza predefinito per"* indica se utilizzare per ogni elemento il proprio colore di sfondo oppure se per tutti gli elementi si deve utilizzare il colore di sfondo definito da Windows.

8.3 Editor

All'interno della pagina Editor si trovano funzioni che riguardano le finestre di codice.



Osservando una finestra di codice, si può notare la presenza di una linea verticale grigia e due orizzontali, anch'esse dello stesso colore:

- La linea verticale è un limite consigliato da non superare: è possibile scegliere se visualizzare questo margine ed anche a quale colonna localizzarlo.
- Le due linee orizzontali servono per segnalare, nella finestra del codice, la fine del file, ed anch'esse sono opzionali.

Premendo il tasto TAB in una finestra di codice, il cursore avanza fino alla tabulazione successiva: la dimensione delle tabulazioni è definibile alla voce *larghezza tabulazi*

Quando si seleziona una parte di testo e si effettua l'operazione di inserimento (vedi utilizzo della clipboard) si puo' notare che il testo evidenziato viene rimpiazzato; per evitare che il testo selezionato scompaia si deve scegliere l'opzione *blocchi consistenti*.

9 API (Interfaccia di programmazione dell'applicazione)

9.1 Introduzione

Con API si intende indicare l'interfaccia di programmazione dell'applicazione ovvero il set di tutte le istruzioni di cui lo sviluppatore può disporre per far fronte alle varie esigenze dell'applicazione stessa, come per esempio creare o leggere file di testo, modificare automaticamente il valore delle porte al verificarsi di particolari eventi o condizioni, inviare mail o SMS, creare rapporti di produzione, importare o salvare ricette, operare sugli oggetti dei templates, eseguire calcoli matematici, eccetera.

Le istruzioni disponibili si trovano in questa guida sotto la categoria *API* raggruppate secondo l'argomento a cui si riferiscono: la categoria "Files" per esempio contiene tutte le istruzioni che si occupano della gestione dei file, mentre la categoria "Gates" tutte le istruzioni che operano sulle porte.

Esempio

```
Function Void Test()
      real v = GetNumGateValue("N",1);
      v = v + 1;
      SetNumGateValue("N",1,v);
end
```

Nella funzione qui sopra, *GetNumGateValue()* e *SetNumGateValue()* sono due istruzioni *API* della categoria *Gates*.
Nel Code Builder, posizionando il cursore sull'istruzione e premendo il tasto F1 è possibile richiamarne automaticamente la guida dettagliata.

'

9.2 Bit

9.2.1 BitAnd

Descrizione
Esegue l'AND bit a bit fra Value1 e Value2

Sintassi
`int BitAnd(int Value1, int Value2)`

Parametri
Value1 primo valore da elaborare
Value2 secondo valore da elaborare

Valore restituito
Risultato dell'operazione Value1 **AND** Value2

Funzioni inerenti
BitOr(),BitXor(),BitNot()

Esempio

```
int Result;
int Value1;
int Value2;
Value1=HexStrToInt("0xFFA7");
Value2=HexStrToInt("0x0FF5");
Result=BitAnd(Value1,Value2);
```

Il risultato è 0xFA5

9.2.2 BitMask

Descrizione
Effettua l'AND bit a bit tra un numero dato ed una maschera

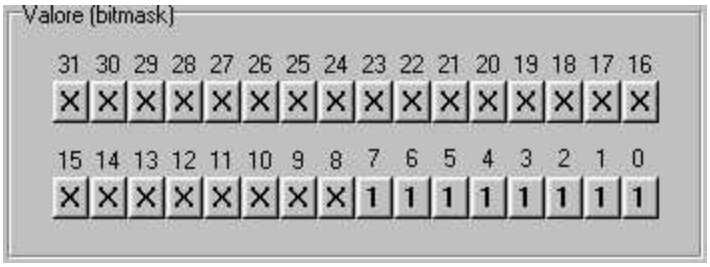
Sintassi
int BitMask(int Num, int Mask)

Parametri
Num numero da manipolare
Mask maschera di bit con cui effettuare l'and

Valore restituito
il numero manipolato con la maschera di bit

Funzioni inerenti
SetBit(), ResetBit(), GetBit()

Esempio
```
LowByte=BitMask(Value,HexStrToInt("000000FF"));
```



9.2.3 BitNot

Descrizione
Esegue la negazione bit a bit del valore ricevuto in ingresso

Sintassi
```
int BitNot(int Value)
```

Parametri
Value valore da elaborare

Valore restituito
Risultato dell'operazione **NOT** Value

Funzioni correlate
BitOr(),BitXor(),BitAnd()

Esempio
```
int Result;
int Value;
Value=HexStrToInt("0xFFA7");
Result=BitNot(Value);
```

Il risultato è 0x0058

9.2.4 BitOr

Descrizione
Esegue l'OR bit a bit fra Value1 e Value2

Sintassi
```
int BitOr(int Value1, int Value2)
```

Parametri
Value1 primo valore da processare
Value2 secondo valore da processare

Valore restituito
Risultato dell'operazione Value1 **OR** Value2

Funzioni inerenti
BitXor(),BitAnd(),BitNot()

Esempio
```
int Result;
int Value1;
int Value2;
Value1=HexStrToInt("0xFFA7");
Value2=HexStrToInt("0x0FF5");
Result=BitOr(Value1,Value2);
```

Il risultato è 0xFFF7

9.2.5 BitXor

Descrizione
Esegue l' XOR bit a bit fra Value1 e Value2

Sintassi
```
int BitXor(int Value1, int Value2)
```

Parametri
Value1 primo valore da elaborare
Value2 secondo valore da elaborare

Valore restituito
Risultato dell'operazione Value1 **XOR** Value2

Funzioni inerenti
BitOr(),BitAnd(),BitNot()

Esempio
```
int Result;
int Value1;
int Value2;
Value1=HexStrToInt("0xFFA7");
Value2=HexStrToInt("0x0FF5");
Result=BitXor(Value1,Value2);
```

Il risultato è 0xF052

9.2.6 GetBit

Descrizione
Restituisce lo stato di un determinato bit

Sintassi
int GetBit(int Num, int Bit)

Parametri
Num numero da manipolare
Bit posizione del bit da controllare (da 0 e 31)

Valore restituito
il bit richiesto

Funzioni inerenti
BitMask(), SetBit(),ResetBit()

Esempio
```
Pari=GetBit(A,0);
```

9.2.7 ResetBit

Descrizione
Imposta un determinato bit di un numero al valore 0

Sintassi
int ResetBit(int Num, int Bit)

Parametri
Num numero da manipolare
Bit posizione del bit da impostare a 0 (da 0 e 31)

Valore restituito
il numero con il bit basso

Funzioni inerenti
BitMask(), SetBit(), GetBit()

Esempio
```
A_Pari=ResetBit(A,0);
```

9.2.8 SetBit

Descrizione
Imposta un determinato bit di un numero al valore 1

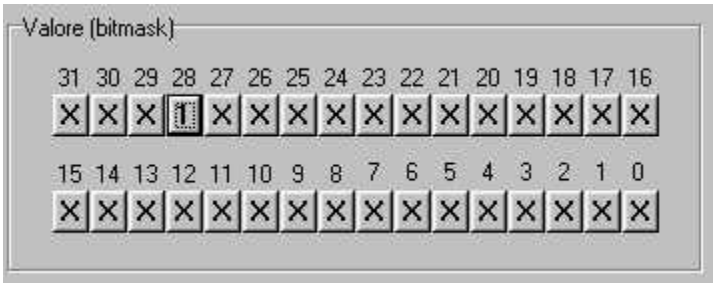
Sintassi
int SetBit(int Num, int Bit)

Parametri
Num numero da manipolare
Bit posizione del bit da impostare a 1 (da 0 e 31)

Valore restituito
il numero con il bit alto

Funzioni inerenti
BitMask(), ResetBit(), GetBit()

Esempio
```
MSB=SetBit(0,28);
```



9.3 Date and Time

9.3.1 DateTimeToSeconds

Descrizione
Converte data e ora ricevuti in ingresso nel numero totale di secondi trascorsi dal 1 Gennaio 1901

Sintassi
```
Unsigned DateTimeToSeconds(int Day,int Month,int Year,int Hour,int
                           Minute,int Second)
```

Parametri

Day	giorno della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.
Month	mese della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.
Year	anno della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.
Hour	ora della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.
Minute	minuti della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.
Seconds	secondi della data ora di cui calcolarne il numero di secondi trascorsi rispetto a 1 Gennaio 1901.

Valore restituito
Numero totale di secondi trascorsi da 1 Gennaio 1901.

Funzioni inerenti
GetDayFromSeconds(),GetMonthFromSeconds(),GetYearFromSeconds(), GetHourFromSeconds(), GetMinuteFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
```

9.3.2 GetDateString

Descrizione
Restituisce una stringa contente la data dove giorni, mesi e anni vengono separati da un carattere d'interpunzione specificato;
è possibile specificare se la data viene rappresentata secondo lo stile europeo oppure no.

Sintassi
String GetDateString(String Separator, Bool EuropeanStyle)

Parametri
Separator il carattere che separa giorni, mesi e anni
EuropeanStyle indica se reppresentare la data secondo il formato europeo

Valore restituito
la data in formato stringa con le specifiche richieste

Funzioni inerenti
GetYear(), GetMonth(), GetDayOfMonths(), GetDayOfWeek()

Esempio
```
CurrentDate=GetDateString(MyFavouriteSeparator,true);
```

9.3.3 GetDayFromSeconds

Descrizione
Restituisce il giorno della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetDayFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero del giorno corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetMonthFromSeconds(),GetYearFromSeconds(), GetHourFromSeconds(), GetMinuteFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Int Day;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
Seconds=Seconds+90000;        // Add 25 hours
Day=GetDayFromSeconds(Seconds);
```

9.3.4 GetDayOfMonth

Descrizione
Restituisce un intero rappresentante il giorno attuale.

Sintassi
Int GetDayOfMonth()

Parametri
-

Valore restituito
il giorno attuale in formato intero

Funzioni inerenti

GetDateString(), GetYear(),GetMonth(), GetDayOfWeek()

Esempio
```
TodayIs=GetDayOfMonth();
```

9.3.5 GetDayOfWeek

Descrizione
Restituisce un intero rappresentante il giorno della settimana (0=Dom, 1=lun, 2=mar, etc).

Sintassi
Int GetDayOfWeek()

Parametri
-

Valore restituito
il giorno della settimana in formato intero

Funzioni inerenti
GetDateString(), GetYear(), GetDayOfMonth(), GetMonth()

Esempio
```
if ( GetDayOfWeek()==0 ) then
      Sunday=true;
end
```

9.3.6 GetHour

Descrizione
Restituisce un intero rappresentante l'ora attuale; minuti, secondi e millisecondi non fanno parte del valore che viene restituito.

Sintassi
Int GetHour()

Parametri
-

Valore restituito
l'ora in formato intero

Funzioni inerenti
GetTimeString(), GetMinute(), GetSecond(), GetMilliSeconds()

Esempio
```
Hour=GetHour();
```

9.3.7 GetHourFromSeconds

Descrizione
Restituisce l'ora della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetHourFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero dell'ora corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetDayFromSeconds(),GetMonthFromSeconds(),GetYearFromSeconds(), GetMinuteFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Int Hour;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
Seconds=Seconds+90000;        // Add 25 hours
Hour=GetHourFromSeconds(Seconds);
```

9.3.8 GetMilliseconds

Descrizione
Restituisce un intero rappresentate il numero di millisecondi dell'ora attuale

Sintassi
Int GetMilliseconds()

Parametri
-

Valore restituito
numero di millisecondi in formato intero

Funzioni inerenti
GetTimeString(), GetHour(), GetMinute(), GetSecond()

Esempio
```
MSecs=GetSecond() / 60 + GetMilliSeconds();
```

9.3.9 GetMinute

Descrizione
Restituisce un intero rappresentate il numero di minuti dell'ora attuale

Sintassi
Int GetMinute()

Parametri
-

Valore restituito
i minuti in formato intero

Funzioni inerenti
GetTimeString(), GetHour(), GetSecond(), GetMilliSeconds()

Esempio
```
Secs=GetMinute()*60;
```

9.3.10 GetMinuteFromSeconds

Descrizione
Restituisce i minuti della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetMinuteFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero dei minuti corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetDayFromSeconds(),GetMonthFromSeconds(),GetYearFromSeconds(), GetHourFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Int Minute;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
Seconds=Seconds+90000;        // Add 25 hours
Minuti=GetMinuteFromSeconds(Seconds);
```

9.3.11 GetMonth

Descrizione
Restituisce un intero rappresentante il mese attuale.

Sintassi
Int GetMonth()

Parametri
-

Valore restituito
il mese attuale in formato intero

Funzioni inerenti
GetDateString(), GetYear(), GetDayOfMonths(), GetDayOfWeek()

Esempio
```
MonthLeft=12-GetMonth();
```

9.3.12 GetMonthFromSeconds

Descrizione
Restituisce il mese della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetMonthFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero del mese corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetDayFromSeconds(),GetYearFromSeconds(), GetHourFromSeconds(), GetMinuteFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Int Month;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
Seconds=Seconds+90000;        // Add 25 hours
Month=GetMonthFromSeconds(Seconds);
```

9.3.13 GetSecond

Descrizione
Restituisce un intero rappresentate il numero di secondi dell'ora attuale

Sintassi
Int GetSecond()

Parametri
-

Valore restituito
i secondi in formato intero

Funzioni inerenti
GetTimeString(), GetHour(), GetMinute(), GetMilliSeconds()

Esempio
```
Secs=GetMinute()*60 + GetSecond();
```

9.3.14 GetSecondFromSeconds

Descrizione
Restituisce i secondi della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetSecondFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero dei secondi corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetDayFromSeconds(),GetMonthFromSeconds(),GetYearFromSeconds(), GetHourFromSeconds(),GetMinuteFromSeconds()

Esempio
```
Unsigned Seconds;
Int Second;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
```

```
Seconds=Seconds+90000;        // Add 25 hours
Second=GetSecondFromSeconds(Seconds);
```

9.3.15 GetTickCount

(OBSOLETA)

Descrizione
Restituisce un intero rappresentante il numero di millisecondi da quando è stato avviato il sistema; Dopo 24.8 giorni circa questo numero riparte da 0.

Sintassi
Int GetTickCount()

Parametri
-

Valore restituito
il numero di millisecondi da quanto il sistema è stato avviato in formato intero

Funzioni inerenti
-

Esempio
```
TimeFromStart = GetTickCount() / TickSize;
```

NB: Si consiglia di utilizzare la funzione UnsignedGetTickCount()

9.3.16 GetTimeString

Descrizione
Restuisce una stringa contenente l'ora dove ora, minuti, secondi e millisecondi vengono separati da un carattere d'interpunzione specificato.

Sintassi
String GetTimeString(String Separator)

Parametri
Separator il carattere che separa ora, minuti e secondi

Valore restituito
l'ora in formato stringa con il separatore richiesto

Funzioni inerenti
GetHour(), GetMinute(), GetSecond(), GetMilliSeconds()

Esempio
```
CurrentTime=GetTimeString(":");
```

9.3.17 GetYear

Descrizione
Restituisce un intero rappresentante l'anno attuale.

Sintassi
Int GetYear()

Parametri
-

Valore restituito
l'anno attuale in formato intero

Funzioni inerenti
GetDateString(), GetMonth(), GetDayOfMonths(), GetDayOfWeek()

Esempio
```
Deltayear=GetYear()-1980;
```

9.3.18 GetYearFromSeconds

Descrizione
Restituisce l'anno della data/ora corrispondente al numero di secondi ricevuto in ingresso calcolati rispetto al 1 Gennaio 1901.

Sintassi
```
Int GetYearFromSeconds(Unsigned Seconds)
```

Parametri
Seconds numero totale di secondi calcolati rispetto al 1 gennaio 1901.

Valore restituito
Numero dell'anno corrispondente.

Funzioni inerenti
DateTimeToSeconds(),GetDayFromSeconds(),GetMonthFromSeconds(), GetHourFromSeconds(), GetMinuteFromSeconds(),GetSecondFromSeconds()

Esempio
```
Unsigned Seconds;
Int Year;
Seconds=DateTimeToSeconds(10,7,1970,5,34,22);
Seconds=Seconds+90000;        // Add 25 hours
Year=GetYearFromSeconds(Seconds);
```

9.3.19 SetDateTime

Descrizione
Imposta la data e l'ora.
Questa istruzione necessita che il Runtime sia eseguito con i diritti di amministratore.

Sintassi
Bool SetDateTime(int Giorno,int Mese,int Anno,int Ore,int Minuti,int Secondi)

Parametri

Giorno	giorno da impostare
Mese	mese da impostare
Anno	anno da impostare
Ore	ore da impostare
Minuti	minuti da impostare
Secondi	secondi da impostare

Valore restituito
True se l'operazione è terminata con successo
False se è avvenuto un errore.

Funzioni inerenti
-

Esempio
```
SetDateTime(10,7,1970,10,11,50);
```

9.3.20 UnsignedGetTickCount

Descrizione
Restituisce un intero rappresentante il numero di millisecondi da quando è stato avviato il sistema; Dopo 49.7 giorni circa questo numero riparte da 0.

Sintassi
Unsigned UnsignedGetTickCount()

Parametri
-

Valore restituito
il numero di millisecondi da quanto il sistema è stato avviato in formato intero senza segno

Funzioni inerenti
-

Esempio
```
Unsigned TimeFromStart;
TimeFromStart = UnsignedGetTickCount();
```

9.4 Devices

9.4.1 DeviceEnableCommunication

Descrizione
Abilita o disabilita la comunicazione con il dispositivo specificato.

Sintassi
```
Bool DeviceEnableCommunication(Int Channel, Int DevNum,Bool ToEnable,Bool
SaveChanges)
```

Parametri
Channel numero del canale
DevNum numero del dispositivo
ToEnable
true (per abilitare la comunicazione con il dispositivo)
false (per disabilitare la comunicazione con il dispositivo)
SaveChanges
se "true" allora la modifica è salvata permanentemente.
se "false" allora la modifica è valida solo per la sessione attuale..

Valore restituito
"true" se la funzione è stata eseguita correttamente
"false" in caso di errore.

Funzioni inerenti
IsDeviceCommunicationEnabled()

Esempio
// Abilita la comunicazione con il dispositivo numero 3 del canale 1
```
DeviceEnableCommunication(1,3,true,true);
```

9.4.2 DeviceName

Descrizione
Restituisce il nome del dispositivo in base al numero di canale ed al numero di dispositivo.

Sintassi
String DeviceName(Int Channel, Int DevNum)

Parametri
Channel numero del canale
DevNum numero del dispositivo

Valore restituito
il nome del dispositivo

Funzioni inerenti
-

Esempio
```
DevName = DeviceName (1,3);
```
// canale 1 terzo dispositivo

9.4.3 GetDeviceRxErrors

Descrizione
Ritorna il numero di errori di comunicazione avvenuti durante tutte le operazioni di lettura dati dal dispositivo indicato.

Sintassi
```
Int GetDeviceRxErrors(Int Channel, Int DevNum)
```

Parametri
Channel numero del canale
DevNum numero del dispositivo

Valore restituito
Numero di errori di comunicazione.
"-1" se è avvenuto un errore durante l'elaborazione dell'istruzione.

Funzioni inerenti
ResetDeviceRxErrors(),GetDeviceTxErrors()

Esempio
```
Int RxErrors;
RxErrors=GetDeviceRxErrors(1,3);
```

9.4.4 GetDeviceTxErrors

Descrizione
Ritorna il numero di errori di comunicazione avvenuti durante tutte le operazioni di scrittura dati verso dispositivo indicato.

Sintassi
```
Int GetDeviceTxErrors(Int Channel, Int DevNum)
```

Parametri
Channel numero del canale
DevNum numero del dispositivo

Valore restituito
Numero di errori di comunicazione.
"-1" se è avvenuto un errore durante l'elaborazione dell'istruzione.

Funzioni inerenti
ResetDeviceTxErrors(),GetDeviceRxErrors()

Esempio
```
Int TxErrors;
TxErrors=GetDeviceTxErrors(1,3);
```

9.4.5 IsDeviceCommunicationEnabled

Descrizione
Ritorna lo stato Abilitato/Disabilitato della comunicazione con il dispositivo indicato.

Sintassi
```
Bool IsDeviceCommunicationEnabled(Int Channel, Int DevNum)
```

Parametri
Channel numero del canale
DevNum numero del dispositivo

Valore restituito
"true" se la comunicazione con il dispositivo è abilitata
"false" se la comunicazione con il dispositivo è disabilitata

Funzioni inerenti
DeviceEnabledCommunication()

Esempio
```
Bool Status;
Status=IsDeviceCommunicationEnabled(1,3);
```

9.4.6 IsDeviceCommunicationKo

Descrizione
Ritorna lo stato di comunicazione con il dispositivo (Ok/Ko).

Sintassi
```
Bool IsDeviceCommunicationKo(Int Channel, Int DevNum)
```

Parametri
Channel numero del canale
DevNum numero del dispositivo

Valore restituito
"true" se la comunicazione con il dispositivo è KO
"false" se la comunicazione con il dispositivo è OK

Funzioni inerenti
IsDeviceCommunicationEnabled()

Esempio
```
Bool ComKo;
ComKo=IsDeviceCommunicationKo(1,3);
```

9.4.7 ResetDeviceRxErrors

Descrizione
Resetta il contatore degli errori di lettura dati dal dispositivo.

Sintassi
```
Bool ResetDeviceRxErrors(Int Channel, Int DevNum)
```

Parametri

Channel	numero del canale
DevNum	numero del dispositivo

Valore restituito
"true" se la funzione è stata eseguita correttamente
"false" in caso di errore.

Funzioni inerenti
GetDeviceRxErrors(),ResetDeviceTxErrors()

Esempio
```
ResetDeviceRxErrors(1,3);
```

9.4.8 ResetDeviceTxErrors

Descrizione
Resetta il contatore degli errori di scrittura dati sul dispositivo.

Sintassi
```
Bool ResetDeviceTxErrors(Int Channel, Int DevNum)
```

Parametri

Channel	numero del canale
DevNum	numero del dispositivo

Valore restituito
"true" se la funzione è stata eseguita correttamente
"false" in caso di errore.

Funzioni inerenti
GetDeviceTxErrors(),ResetDeviceRxErrors()

Esempio
```
ResetDeviceTxErrors(1,3);
```

9.5 Directory

9.5.1 DirectoryCreate

Descrizione
Crea una nuova directory

Sintassi
Bool DirectoryCreate(String DirName)

Parametri
DirName nome della nuova directory

Valore restituito
true (se la directory viene creata)
false (in caso di fallimento)

Funzioni inerenti
DirectoryDelete()

Esempio
```
DirectoryCreate(GetProjectPath() + "\MyDirectory");
```

9.5.2 DirectoryDelete

Descrizione
Elimina una directory: se essa non è vuota, non verrà eliminata.

Sintassi
Bool DirectoryDelete(String DirName)

Parametri
DirName nome della directory da eliminare

Valore restituito
true (se la directory viene eliminata)
false (in caso di fallimento)

Funzioni inerenti
DirectoryCreate()

Esempio
```
DirectoryDelete(GetProjectPath() + "\MyDirectory");
```

9.5.3 DirectoryGetCurrent

Descrizione
Restituisce il percorso attuale

Sintassi
String DirectoryGetCurrent()

Parametri
-

Valore restituito
una stringa contenente la directory corrente

Funzioni inerenti
DirectorySetCurrent()

Esempio
```
SearchFor=DirectoryGetCurrent()+"\*.*";
```

9.5.4 DirectorySetCurrent

Descrizione
Definisce il percorso attuale

Sintassi
Bool DirectorySetCurrent(String NewPath)

Parametri
NewPath nuovo percorso

Valore restituito
true (se la directory viene cambiata)
false (in caso di fallimento)

Funzioni inerenti
DirectoryGetCurrent()

Esempio
```
DirectorySetCurrent("y:\projects");
```

9.6 Files

9.6.1 FileAttrFound

(OBSOLETA)
Si consiglia di utilizzare la funzione FileAttrFoundEx.

Descrizione
Restituisce l'attributo del file letto con *FileFindFirst* o *FileFindNext*

Sintassi
Int FileAttrFound()

Parametri
-

Valore restituito
l'attributo del file letto

Funzioni inerenti
FileFindNext(), FileFindClose(), FileFindFirst(), FileNameFound()

Esempio
```
Function void exampleFindFirst()
        String CurrPath;
        String CurrFile;
        CurrPath=DirectoryGetCurrent()+"\*.*";
        if (FileFindFirst(CurrPath)==0) then
                CurrFile=FileNameFound();
                MessageBox(CurrFile,CurrPath);
                while (FileFindNext() == 0)
                        CurrFile=FileNameFound();
                        MessageBox(CurrFile,CurrPath);
                end
        end
        FileFindClose();
End
```

9.6.2 FileAttrFoundEx

Descrizione
Restituisce l'attributo del file letto con *FileFindFirstEx* o *FileFindNextEx*

Sintassi
Int FileAttrFoundEx(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
l'attributo del file letto

Funzioni inerenti
FileFindFirstEx(), FileFindNextEx(), FileFindCloseEx(), FileNameFoundEx()

Esempio
```
Function void FindFile()
	String CurrPath;
	String CurrFile;
	int Handle;
	CurrPath=GetProjectPath()+"\CSV\*.csv";
	Handle=FileFindFirstEx(CurrPath);
	if (Handle!=0) then
		CurrFile=FileNameFoundEx(Handle);
		if (CurrFile!="") then
			MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			while (FileFindNextEx(Handle)!=false)
				CurrFile=FileNameFoundEx(Handle);
				MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			end
		end
		FileFindCloseEx(Handle);
	end
end
```

9.6.3 FileCopy

Descrizione
Copia un file

Sintassi
Bool FileCopy(String Src, String Dst, Bool IfExists)

Parametri
Src nome del file da copiare
Dst nuovo nome del file
IfExists true (se Dst esiste non viene sovrascritto) false (se Dst esiste viene sovrascritto)

Valore restituito
true (nel caso l'operazione venga effettuata con successo)
false (nel caso l'operazione fallisca)

Funzioni inerenti
FileMove()

Esempio
```
if (FileExist("C:\Temporan\Promem.txt")) then
```

```
      FileCopy("C.\Temporan\Promem.txt","C:\Temporan\Promem.bak",false);
//overwrite if exists
end
```

9.6.4 FileClose

Descrizione
Chiude un file aperti in precedenza

Sintassi
int FileClose(Int Handle)

Parametri
Handle gestore del file da chiudere

Valore restituito
0 in caso di successo (diverso da 0 in caso di fallimento)

Funzioni inerenti
FileOpen()

Esempio
```
FileHandle=FileOpen("C:\Temporan\Starter.dat","rt"); // apre il file di
testo in lettura
FileWriteLn(FileHandle,"Stringa di prova");
FileWriteLn(FileHandle,"questa è la seconda riga");
FileClose(FileHandle);
```

9.6.5 FileDelete

Descrizione
Cancella un file

Sintassi
Bool FileDelete(String FileName)

Parametri
FileName nome del file

Valore restituito
true (in caso di operazione eseguita con successo)
false (in caso di errore)

Funzioni inerenti
-

Esempio
```
if (FileExist("C:\Temporan\Promem.bak")) then
      FileDelete("C:\Temporan\Promem.bak");
end
```

9.6.6 FileEof

Descrizione
restituisce un valore diverso da 0 se il puntatore di lettura/scrittura ha raggiunto la fine del file

Sintassi
Int FileEOF(Int Handle)

Parametri
Handle gestore del file

Valore restituito
un valore diverso da 0 quando la fine del file è stata raggiunta

Funzioni inerenti
-

Esempio
```
int i=0;
FileHandle=FileOpen("C:\Temporan\MadeByAPI.dat","rb"); // apre il file
binario in lettura

While (FileEOF(FileHandle) == 0) // leggi 3 caratteri per volta
  i=i+1;
  s=FileRead(FileHandle,3);
  if (FileEOF(FileHandle) == 0) then           // controlla che il dato
sia valido
       MessageBox(s,"terzina numero "+IntToStr(i));
  end
End
FileClose(FileHandle);
```

9.6.7 FileExist

Descrizione
Determina l'esistenza di un file

Sintassi
Bool FileExist(String FileName)

Parametri
FileName nome del file

Valore restituito
true (nel caso il file esista)
false (se il file non esiste)

Funzioni inerenti
-

Esempio
```
if (FileExist("C:\Temporan\Promem.bak")) then
      FileDelete("C:\Temporan\Promem.bak");
end
```

9.6.8 FileFindClose

(OBSOLETA)
Si consiglia di utilizzare la funzione FileFindCloseEx().

Descrizione

Termina l'operazione di lettura iniziato con *FileFindFirst* e libera le risorse allocate a tale scopo

Sintassi
void FileFindClose()

Parametri
-

Valore restituito
-

Funzioni inerenti
FileFindNext(), FileFindFirst(), FileNameFound(), FileAttrFound()

Esempio
```
Function void exampleFindFirst()
	String CurrPath;
	String CurrFile;
	CurrPath=DirectoryGetCurrent()+"\*.*";
	if (FileFindFirst(CurrPath)==0) then
		CurrFile=FileNameFound();
		MessageBox(CurrFile,CurrPath);
		while (FileFindNext() == 0)
			CurrFile=FileNameFound();
			MessageBox(CurrFile,CurrPath);
		end
	end
	FileFindClose();
End
```

9.6.9 FileFindCloseEx

Descrizione
Termina l'operazione di lettura iniziato con *FileFindFirstEx* e libera le risorse allocate a tale scopo

Sintassi
void FileFindCloseEx(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
-

Funzioni inerenti
FileFindFirstEx(), FileFindNextEx(), FileNameFoundEx(), FileAttrFoundEx()

Esempio
```
Function void FindFile()
	String CurrPath;
	String CurrFile;
	int Handle;
	CurrPath=GetProjectPath()+"\CSV\*.csv";
	Handle=FileFindFirstEx(CurrPath);
	if (Handle!=0) then
		CurrFile=FileNameFoundEx(Handle);
		if (CurrFile!="") then
			MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			while (FileFindNextEx(Handle)!=false)
				CurrFile=FileNameFoundEx(Handle);
				MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			end
```

```
                end
                FileFindCloseEx(Handle);
        end
end
```

9.6.10 FileFindFirst

(OBSOLETA)
Si consiglia di utilizzare la funzione FileFindFirstEx().

Descrizione
Inizia la lettura dei file di una directory

Sintassi
Bool FileFindFirst(String Path)

Parametri
Path percorso da cui leggere la lista dei files

Valore restituito
true (se esiste almeno un file)
false (se la directory è vuota)

Funzioni inerenti
FileFindNext(), FileFindClose(), FileNameFound(), FileAttrFound()

NB: NON eseguire più *findfirst* in contemporanea, una volta iniziata una sequenza la si deve portare a termine col *FindClose* per liberare le risorse utilizzate

Esempio

```
Function void exampleFindFirst()
        String CurrPath;
        String CurrFile;
        CurrPath=DirectoryGetCurrent()+"\*.*";
        if (FileFindFirst(CurrPath)==0) then
                CurrFile=FileNameFound();
                MessageBox(CurrFile,CurrPath);
                while (FileFindNext() == 0)
                        CurrFile=FileNameFound();
                        MessageBox(CurrFile,CurrPath);
                end
        end
        FileFindClose();
End
```

9.6.11 FileFindFirstEx

Descrizione
Inizia la lettura dei file di una directory

Sintassi
int FileFindFirstEx(String Path)

Parametri
Path percorso da cui leggere la lista dei files

Valore restituito
Handle asseganto gestore dell'operazione di ricerca in corso. Questo handle deve essere successivamente usato anche nelle istruzioni FileFindNextEx() e FileFindCloseEx().

Per sapere se è stato trovato almeno un file controllare che FileNameFoundEx() restituisca una stringa diversa da "".

Funzioni inerenti
FileFindNextEx(), FileNameFoundEx(), FileAttrFoundEx(), FileFindCloseEx()

Esempio
```
Function void FindFile()
	String CurrPath;
	String CurrFile;
	int Handle;
	CurrPath=GetProjectPath()+"\CSV\*.csv";
	Handle=FileFindFirstEx(CurrPath);
	if (Handle!=0) then
		CurrFile=FileNameFoundEx(Handle);
		if (CurrFile!="") then
			MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			while (FileFindNextEx(Handle)!=false)
				CurrFile=FileNameFoundEx(Handle);
				MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			end
		end
		FileFindCloseEx(Handle);
	end
end
```

9.6.12 FileFindNext

(OBSOLETA)
Si consiglia di utilizzare la funzione FileFindNextEx().

Descrizione
Continua la lettura dei file di una directory

Sintassi
Bool FileFindNext()

Parametri
-

Valore restituito
true (se è stato letto un file)
false (non ci sono più file da leggere)

Funzioni inerenti
FileFindFirst(), FileFindClose(), FileNameFound(), FileAttrFound()

Esempio
```
Function void exampleFindFirst()
	String CurrPath;
	String CurrFile;
	CurrPath=DirectoryGetCurrent()+"\*.*";
	if (FileFindFirst(CurrPath)==0) then
		CurrFile=FileNameFound();
		MessageBox(CurrFile,CurrPath);
```

```
                while (FileFindNext() == 0)
                        CurrFile=FileNameFound();
                        MessageBox(CurrFile,CurrPath);
                end
        end
        FileFindClose();
End
```

9.6.13 FileFindNextEx

Descrizione
Continua la lettura dei file di una directory

Sintassi
bool FileFindNextEx(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
true (se è stato letto un file)
false (non ci sono più file da leggere)

Funzioni inerenti
FileFindFirstEx(), FileNameFoundEx(), FileAttrFoundEx(), FileFindCloseEx()

Esempio

```
Function void FindFile()
        String CurrPath;
        String CurrFile;
        int Handle;
        CurrPath=GetProjectPath()+"\CSV\*.csv";
        Handle=FileFindFirstEx(CurrPath);
        if (Handle!=0) then
                CurrFile=FileNameFoundEx(Handle);
                if (CurrFile!="") then
                        MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
                        while (FileFindNextEx(Handle)!=false)
                                CurrFile=FileNameFoundEx(Handle);
                                MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
                        end
                end
                FileFindCloseEx(Handle);
        end
end
```

9.6.14 FileGetAttr

Descrizione
Restituisce gli attributi di un file

Sintassi
Int FileGetAttr(String FileName)

Parametri
FileName nome del file

Valore restituito
attributo attuale del file

Funzioni inerenti

FileSetAttr()

Esempio

```
// forza l'attributo di sola lettura
FileSetAttr("C:\Temporan\TryMe.bat",FileGetAttr("C:\Temporan\TryMe.bat")|1)
;
```

9.6.15 FileGetSize

Descrizione
Restituisce la dimensione del file

Sintassi
Int FileGetSize(String FileName)

Parametri
FileName nome del file

Valore restituito
dimensione del file in bytes

Funzioni inerenti
-

Esempio

```
CCDim=FileGetSize("C:\Temporan\Text.txt");
```

9.6.16 FileMove

Descrizione
Sposta un file

Sintassi
Bool FileMove(String From, String To)

Parametri
From nome del file da spostare
To nuovo destinazione

Valore restituito
true (nel caso l'operazione venga effettuata con successo)
false (nel caso l'operazione fallisca)

Funzioni inerenti
FileCopy()

Esempio

```
if (FileExist("C:\Temporan\Promem.txt")) then
      FileMove("C:\Temporan\Promem.txt","E:\Temp\Promem.txt");
end
```

9.6.17 FileNameFound

(OBSOLETA)

Si consiglia di utilizzare la funzione FileNameFoundEx().

Descrizione
Restituisce il nome del file letto con *FileFindFirst* o *FileFindNext*

Sintassi
String FileNameFound()

Parametri
-

Valore restituito
il nome del file letto

Funzioni inerenti
FileFindNext(), FileFindClose(), FileFindFirst()

Esempio
```
Function void exampleFindFirst()
	String CurrPath;
	String CurrFile;
	CurrPath=DirectoryGetCurrent()+"\*.*";
	if (FileFindFirst(CurrPath)==0) then
		CurrFile=FileNameFound();
		MessageBox(CurrFile,CurrPath);
		while (FileFindNext() == 0)
			CurrFile=FileNameFound();
			MessageBox(CurrFile,CurrPath);
		end
	end
	FileFindClose();
End
```

9.6.18 FileNameFoundEx

Descrizione
Restituisce il nome del file letto con *FileFindFirstEx* o *FileFindNextEx*

Sintassi
String FileNameFoundEx(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
il nome del file letto

Funzioni inerenti
FileFindFirstEx(), FileNameFoundEx(), FileAttrFoundEx(), FileFindCloseEx()

Esempio
```
-Function void FindFile()
	String CurrPath;
	String CurrFile;
	int Handle;
	CurrPath=GetProjectPath()+"\CSV\*.csv";
	Handle=FileFindFirstEx(CurrPath);
	if (Handle!=0) then
		CurrFile=FileNameFoundEx(Handle);
		if (CurrFile!="") then
			MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
			while (FileFindNextEx(Handle)!=false)
```

```
                        CurrFile=FileNameFoundEx(Handle);
                        MessageBox("File Foud: "+CurrFile,"FileNameFoudEx");
                    end
                end
                FileFindCloseEx(Handle);
            end
        end
```

9.6.19 FileOpen

Descrizione
Apre un file nella modalità specificata

Sintassi
int FileOpen(String FileName, String Mode)

Parametri
FileName nome del file
Mode modalità di apertura

Valore restituito
il gestore del file in caso di successo, 0 in caso di errore

Funzioni inerenti
FileClose()

Esempio
```
FileHandle=FileOpen("C:\Temporan\Promem.txt","wt"); // apre il file di
testo in scrittura
FileWriteLn(FileHandle,"Stringa di prova");
FileWriteLn(FileHandle,"questa è la seconda riga");
FileClose(FileHandle);
```

Modalità di apertura:

Valore	Descrizione
r	Apre il file in sola lettura
w	Apre il file in scrittura se non esiste lo crea e se esiste lo sovrascrive
a	Apre il file aggiungendo le informazioni a quelle presenti se non esiste lo crea
r+	Apre un file per l'aggiornamento (lettura e scrittura))
w+	Crea un nuovo file pronto per l'utilizzo in lettura e scrittura.Se il file esiste viene sovrascritto.
a+	Apre il file per l'aggiunta di informazioni o lo crea se non esiste e lo prepara per l'utilizzo in scrittura e lettura

Aggiungendo *t* oppure *b* nella stringa si sceglie di aprire il file rispettivamente in modalità *testo o* in modalità *binaria*.

9.6.20 FilePos

Descrizione
restituisce la posizione corrente del puntatore di lettura/scrittura di un file aperto

Sintassi
Int FilePos(Int Handle)

Parametri

Handle gestore del file

Valore restituito
Posizione del puntatore di lettura/scrittura

Funzioni inerenti
FileSeek()

Esempio
```
CurPos=FilePos(CurrHandle);
```

9.6.21 FileRead

Descrizione
Legge dei caratteri da un file binario.

Sintassi
String FileRead(Int Handle, Int Length)

Parametri
Handle gestore del file da cui leggere
Length numero di caratteri da leggere

Valore restituito
I caratteri letti dal file

Funzioni inerenti
FileWrite(), FileSize()

Esempio
```
Buffer=FileRead(FileHandle,16); // legge 16 caratteri dal file
```

9.6.22 FileReadLn

Descrizione
Legge una linea da un file di testo.

Sintassi
String FileReadLn(Int Handle)

Parametri
Handle gestore del file di testo da cui leggere

Valore restituito
La linea letta

Funzioni inerenti
FileWriteLn(), FileOEF()

Esempio
```
CurrLine=FileReadLn(TextFileHandle);
```

9.6.23 FileRename

Descrizione
Rinomina un file

Sintassi
Int FileRename(String OldName, String NewName)

Parametri
OldName nome del file da rinominare
NewName nuovo nome del file

Valore restituito
0 (nel caso l'operazione venga eseguita con successo)
-1 (in caso di errore)

Funzioni inerenti
-

Esempio
```
if (FileExist("C:\Temporan\Promem.bak")) then
      FileRename("C:\Temporan\Promem.bak","Promem.txt");
end
```

9.6.24 FileSeek

Descrizione
Posiziona il puntatore di lettura/scrittura di un file aperto

Sintassi
int FileSeek(Int Handle, Int Offset, Int Whence)

Parametri
Handle il gestore del file su cui lavorare
Offset quanto spostare il puntatore
Whence riferimento da cui effettuare lo spostamento

Valore restituito
0 in caso di successo (diverso da 0 in caso di fallimento)

Funzioni inerenti
FilePos()

Esempio
`FileSeek (FileHandle, 15,0);` // posizione 15 dall'inizio del file

Riferimento (parametro Whence)

Valore	Descrizione
0	posizione relativa all'inizio del file
1	posizione relativa a quella attuale
2	posizione relativa alla fine del file

9.6.25 FileSetAttr

Descrizione
Definisce gli attributi di un file

Sintassi
Int FileGetAttr(String FileName,Int NewAttr)

Parametri
FileName nome del file
NewAttr i nuovi attributi del file

Valore restituito
0 (in caso gli attributi vengano modificati)
valore non nullo (in caso di errore)

Funzioni inerenti
FileGetAttr()

Esempio

```
// forza l'attributo di sola lettura
FileSetAttr("C:\Temporan\TryMe.bat",FileGetAttr("C:\Temporan\TryMe.bat")|1)
;
```

9.6.26 FileSize

Descrizione
restituisce la dimensione di un file aperto

Sintassi
Int FileSize(Int Handle)

Parametri
Handle gestore del file

Valore restituito
dimensione del file

Funzioni inerenti
-

Esempio

```
DirSize=DirSize+FileSize(CurrFileHandle);
```

9.6.27 FileSplitPath

Descrizione
Permette di accedere agli elementi di un percorso file separatamente

Sintassi
String FileSplitPath(String FileName, Int Elem)

Parametri
FileName nome del file completo di percorso
Elem l'elemento di FileName richiesto:
0 - Drive
1 - Percorso
2 - Nome del file
3 - Estensione

Valore restituito
l'elemento richiesto

Funzioni inerenti
-

Esempio
```
String FN="c:\windows\system\win.ini";
String Drive;
String Path;
String FileName;
String Ext;

Drive=FileSplitPath(FN,0);   // C
Path=FileSplitPath(FN,1);   // \WINDOWS\SYSTEM
FileName=FileSplitPath(FN,2);   // WIN
Ext=FileSplitPath(FN,3); // .ini
```

9.6.28 FileWrite

Descrizione
Scrive dei caratteri su un file binario.

Sintassi
int FileWrite(Int Handle, String Data, Int Length)

Parametri
Handle gestore del file su cui scrivere
Data i caratteri da scrivere
Length numero di caratteri che si vuole scrivere

Valore restituito
Il numero di caratteri che sono stati scritti sul file

Funzioni inerenti
FileRead(), FileSize()

Esempio
```
FileWrite(FileHandle,Buffer,16); // scrive 16 caratteri dal Buffer nel file
```

9.6.29 FileWriteLn

Descrizione
Scrive una linea in un file di testo.

Sintassi
Int FileWriteLn(Int Handle, String Line)

Parametri
Handle gestore del file di testo su cui scrivere
Line stringa da scrivere

Valore restituito
un valore non negativo in caso di successo

Funzioni inerenti
FileReadLn(), FileOEF()

Esempio
```
FileWriteLn(TextFileHandle,UserInserted);
```

9.7 Gates

9.7.1 NumGates

9.7.1.1 GetNumGateCommunicationStatus

Descrizione:
Ritorna lo stato di comunicazione della porta numerica specificata.

Sintassi:
Bool GetNumGateCommunicationStatus(string Name,int Id)

Parametri:
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta non è in errore di comunicazione;
False= la porta non è stata trovata o è in errore di comunicazione

Funzioni inerenti:
GetNumGateValue(),SetNumGateValue(),GetNumGateProp()

Esempio:
```
...
if(GetNumGateCommunicationStatus(NUM0,ID0)) then
  GetNumGateValue(NUM0,ID0);
  ...
end
...
```

9.7.1.2 GetNumGateGateID

Descrizione
Restituisce il nome (campo "*ID Porta*" in *GateBuilder*) della porta numerica indicata.

Sintassi
String GetNumGateGateID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte numeriche.

Valore restituito
GateID sotto forma di stringa

Funzioni inerenti
GetNumGateNID()

Esempio
```
string GateID;
GateID = GetNumGateGateID (10);
```

9.7.1.3 GetNumGateNID

Descrizione
Restituisce l'identificatore (campo "*N ID*" in *GateBuilder*) della porta numerica indicata.

Sintassi
Int GetNumGateNID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte numeriche.

Valore restituito
Gate NID sotto forma di intero

Funzioni inerenti
GetNumGateGateID()

Esempio
```
int GateNID;
GateNID = GetNumGateNID (10);
```

9.7.1.4 GetNumGateProp

Descrizione
Restituisce una proprietà della porta numerica sotto forma di stringa.

Sintassi
String GetNumGateProp(String Name, Int Id, int Property)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)
Property numero della proprietà che si vuole ottenere

PROPRIEA'	COSTANTE NUMERICA
numero del canale	0
numero del dispositivo	1
indirizzo della porta	2
tipo di valore	3
numero di decimali (approssimazione)	4
valore minimo	5
valore massimo	6
fattore moltiplicativo	7
fattore somma	8
tolleranza	9
frequenza di campionamento	10
stato lettura porta	11
stato scrittura porta	12

Valore restituito
proprietà richiesta sotto forma di stringa

Funzioni inerenti
-

Esempio
samplingRate = GetNumGateProp (NUM1, ID1, 10); // frequenza di campionamento!

9.7.1.5 GetNumGateValue

Descrizione
Restituisce il valore della porta specificata.

Sintassi
Real GetNumGateValue(String Name, Int Id)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito
valore della porta specificata

Funzioni inerenti
SetNumGateValue()

Esempio
int Gate12Id;
String Gate12Name;
Gate12 = GetNumGateValue(Gate12Name, Gate12Id);

9.7.1.6 GetNumGateValueAsString

Descrizione
Restituisce il valore della porta specificata sottoforma di testo formattato.

Sintassi
String GetNumGateValueAsString(String Name, Int Id,String Format)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Format formato del testo da restituire.
Utilizzando il formato “%g” si visualizza il numero reale in modo tale che occupi il minor spazio possibile (eventualmente viene utilizzata la forma esponenziale). Se si desidera indicare come visualizzare il numero è possibile usare la notazione “**%**x**.**y**lf**”, dove:

·**x** è un numero (opzionale) e indica il numero di cifre totali da visualizzare. Se manca verranno visualizzare tutte le cifre del valore letto dalla porta, mentre se è preceduto da 0 verranno visualizzati degli 0 prima del numero fino a raggiungere il numero di cifre indicato.
·**y** è anch'esso un numero (opzionale) e indica il numero di cifre da visualizzare dopo la virgola.

Se Y è uguale a "*" allora viene usato come numero di cifre decimali quello specificato per la porta nel Gate Builder.
Ecco di seguito alcuni esempi:
“%5.2lf” produrrà 123.45
“%5.0lf” produrrà 123
“%07.2lf” produrrà 00123.45
"%7.*lf" produrrà 123.456 se il numero di cifre decimali specificato per la porta associata (nel Gate Builder) è 3.

In modo analogo è possibile specificare il formato di visualizzazione per i numeri interi

("%xd") . Il significato del parametro x è identico a quello visto sopra.

Valore restituito
valore della porta specificata sottofroma di testo

Funzioni inerenti
GetNumGateValue()

Esempio
```
int Gate12Id;
String Gate12Name;
String Value;
Value = GetNumGateValueAsString(Gate12Name, Gate12Id,"%7.*lf");
```

9.7.1.7 GetTotalNumGates

Descrizione
Restituisce il numero di porte numeriche definite nell'applicazione.

Sintassi
Int GetTotalNumGates()

Parametri
-

Valore restituito
Un intero contenente il numero di porte numeriche definite nell'applicazione

Funzioni inerenti
GetTotalDigGates(),GetTotalEvnGates(),GetTotalCmpGates(),GetTotalStrGates()

Esempio
```
TotalGates=GetTotalNumGates();
```

9.7.1.8 NumGateExists

Descrizione:
Verifica se la porta numerica indicata è definita.

Sintassi:
Bool NumGateExists(string Name,int Id)

Parametri:
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta è definita nell'applicazione
False= la porta non è definita nell'applicazione

Funzioni inerenti:
CmpGateExists() , DigGateExists() , EvnGateExists() , StrGateExists()

Esempio:
```
Bool ok;
ok = NumGateExists("N",1);
```

9.7.1.9 SetNumGateInMonitor

Descrizione
Abilita/disabilita il campionamento delle porte numeriche definite da campionare "Se in schema".

Sintassi
Bool SetNumGateInMonitor(String Name, Int Id, Bool Enable)

Parametri

Name	nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id	identificatore della porta (campo "*N ID*" in *GateBuilder*)
Enable	"true" abilita il campionamento della porta , "false" disabilita il campionamento della porta.

Valore restituito
true (se la porta esiste)
false (in caso contrario)

Funzioni inerenti
SetDigGateInMonitor(),SetStrGateInMonitor()

Esempio
SetNumGateInMonitor(PrimaryName,PrimaryID,true);

9.7.1.10 SetNumGateValue

Descrizione
Modifica il valore della porta specificata.

Sintassi
Bool SetNumGateValue(String Name, Int Id, Real Value)

Parametri

Name	nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id	identificatore della porta (campo "*N ID*" in *GateBuilder*)
Value	valore numerico che si vuole assegnare alla porta

Valore restituito
true (se la porta esiste ed è stata modificata)
false (in caso contrario)

Funzioni inerenti
GetNumGateValue()

Esempio
```
real Value = 123;
SetNumGateValue("SetPoint",1, Value);
```

9.7.2 DigGates

9.7.2.1 DigGateExists

Descrizione:
Verifica se la porta digitale indicata è definita.

Sintassi:
Bool DigGateExists(string Name,int Id)

Parametri:

Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta è definita nell'applicazione
False= la porta non è definita nell'applicazione

Funzioni inerenti:
CmpGateExists() , EvnGateExists() , NumGateExists() , StrGateExists()

Esempio:
```
Bool ok;
ok = DigGateExists("D",1);
```

9.7.2.2 GetDigGateCommunicationStatus

Descrizione:
Ritorna lo stato di comunicazione con la porta digitale specificata.

Sintassi:
Bool GetDigGateCommunicationStatus(string Name,int Id)

Parametri:
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta non è in errore di comunicazione;
False= la porta non è stata trovata o è in errore di comunicazione

Funzioni inerenti:
GetDigGateValue(),SetDigGateValue()

Esempio:
```
...
if(GetDigGateCommunicationStatus(DIG1,ID1)) then
   GetDigGateValue(DIG1,ID1);
   ...
end
...
```

9.7.2.3 GetDigGateGateID

Descrizione
Restituisce il nome (campo "*ID Porta*" in *GateBuilder*) della porta digitale indicata.

Sintassi
String GetDigGateGateID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte digitali.

Valore restituito
GateID sotto forma di stringa

Funzioni inerenti
GetDigGateNID()

Esempio
```
string GateID;
GateID = GetDigGateGateID (10);
```

9.7.2.4 GetDigGateNID

Descrizione
Restituisce l'identificatore (campo "*N ID*" in *GateBuilder*) della porta digitale indicata.

Sintassi
Int GetDigGateNID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte digitali.

Valore restituito
Gate NID sotto forma di intero

Funzioni inerenti
GetDigGateGateID()

Esempio
```
int GateNID;
GateNID = GetDigGateNID (10);
```

9.7.2.5 GetDigGateProp

Descrizione
Restituisce una proprietà della porta digitale sotto forma di stringa.

Sintassi
String GetDigGateProp(String Name, Int Id, int Property)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)
Property numero della proprietà che si vuole ottenere

PROPRIETA'	COSTANTE NUMERICA
numero del canale	0
numero del dispositivo	1
indirizzo della porta	2
frequenza di campionamento	10
stato lettura porta	11
stato scrittura porta	12

Valore restituito
proprietà richiesta sotto forma di stringa

Funzioni inerenti
-
Esempio
```
samplingRate = GetDigGateProp (DIG1, ID1, 10); // frequenza di campionamento!
```

9.7.2.6 GetDigGateValue

Descrizione
Restituisce il valore intero della porta digitale specificata.

Sintassi
Int GetDigGateValue(String Name, Int Id)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito
valore (0 o 1) della porta specificata .

Funzioni inerenti
SetDigGateValue()

Esempio

```
DigiGate0001= GetDigGateValue("Gate0001",1);
```

9.7.2.7 GetTotalDigGates

Descrizione
Restituisce il numero di porte digitali definite nell'applicazione.

Sintassi
Int GetTotalDigGates()

Parametri
-

Valore restituito
Un intero contenente il numero di porte digitali definite nell'applicazione

Funzioni inerenti
GetTotalCmpGates(),GetTotalEvnGates(),GetTotalNumGates(),GetTotalStrGates()

Esempio

```
TotalGates=GetTotalDigGates();
```

9.7.2.8 SetDigGateInMonitor

Descrizione
Abilita/disabilita il campionamento delle porte digitali definite da campionare "Se in schema".

Sintassi
Bool SetDigGateInMonitor(String Name, Int Id, Bool Enable)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)
Enable "true" abilita il campionamento della porta , "false" disabilita il campionamento della porta.

Valore restituito
true (se la porta esiste)
false (in caso contrario)

Funzioni inerenti
SetNumGateInMonitor(),SetStrGateInMonitor()

Esempio
```
SetDigGateInMonitor(PrimaryName,PrimaryID,true);
```

9.7.2.9 SetDigGateValue

Descrizione
Modifica il valore della porta digitale specificata.

Sintassi
Bool SetDigGateValue(String Name, Int Id, Int Value)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)
Value nuovo valore da assegnare alla porta (0 o 1)

Valore restituito
true (se la porta esiste ed è stata modificata)
false (in caso contrario)

Funzioni inerenti
GetDigGateValue()

Esempio
```
SetDigGateValue("Flag",1, 1);
```

9.7.3 CmpGates

9.7.3.1 CmpGateExists

Descrizione:
Verifica se la porta composta indicata è definita.

Sintassi:
Bool CmpGateExists(string Name,int Id)

Parametri:
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta è definita nell'applicazione
False= la porta non è definita nell'applicazione

Funzioni inerenti:
DigGateExists() , EvnGateExists() , NumGateExists() , StrGateExists()

Esempio:
```
Bool ok;
ok = CmpGateExists("C",1);
```

9.7.3.2 GetCmpGateGateID

Descrizione
Restituisce il nome (campo "*ID Porta*" in *GateBuilder*) della porta composta indicata.

Sintassi

String GetCmpGateGateID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte composte.

Valore restituito
GateID sotto forma di stringa

Funzioni inerenti
GetCmpGateNID()

Esempio
```
string GateID;
GateID = GetCmpGateGateID (10);
```

9.7.3.3 GetCmpGateNID

Descrizione
Restituisce l'identificatore (campo "*N ID*" in *GateBuilder*) della porta composta indicata.

Sintassi
Int GetCmpGateNID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte composte.

Valore restituito
Gate NID sotto forma di intero

Funzioni inerenti
GetCmpGateGateID()

Esempio
```
int GateNID;
GateNID = GetCmpGateNID (10);
```

9.7.3.4 GetCmpGateValue

Descrizione
Restituisce il valore della porta complessa specificata.

Sintassi
Real GetCmpGateValue(String Name, Int Id)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito
valore della porta specificata

Funzioni inerenti
-

Esempio
```
String PortName;
GateName="GateA";
CurrValue= GetCmpGateValue(GateName,GateNo);
```

9.7.3.5 GetCmpGateValueAsString

Descrizione
Restituisce il valore della porta specificata.

Sintassi
String GetCmpGateValueAsString(String Name, Int Id,String Format)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Format formato del testo da restituire.
Utilizzando il formato “%g” si visualizza il numero reale in modo tale che occupi il minor spazio possibile (eventualmente viene utilizzata la forma esponenziale). Se si desidera indicare come visualizzare il numero è possibile usare la notazione “**%**x**.**y**lf**”, dove:

·**x** è un numero (opzionale) e indica il numero di cifre totali da visualizzare. Se manca verranno visualizzare tutte le cifre del valore letto dalla porta, mentre se è preceduto da 0 verranno visualizzati degli 0 prima del numero fino a raggiungere il numero di cifre indicato.
·**y** è anch'esso un numero (opzionale) e indica il numero di cifre da visualizzare dopo la virgola.

Se Y è uguale a "*" allora viene usato come numero di cifre decimali quello specificato per la porta nel Gate Builder.
Ecco di seguito alcuni esempi:
“%5.2lf” produrrà 123.45
“%5.0lf” produrrà 123
“%07.2lf” produrrà 00123.45
"%7.*lf" produrrà 123.456 se il numero di cifre decimali specificato per la porta associata (nel Gate Builder) è 3.

In modo analogo è possibile specificare il formato di visualizzazione per i numeri interi (“%xd”) . Il significato del parametro x è identico a quello visto sopra.

Valore restituito
valore della porta specificata sottoforma di testo

Funzioni inerenti
GetCmpGateValue()

Esempio
```
int Gate12Id;
String Gate12Name;
String Value;
Value = GetCmpGateValueAsString(Gate12Name, Gate12Id,"%7.*lf");
```

9.7.3.6 GetTotalCmpGates

Descrizione
Restituisce il numero di porte composte definite nell'applicazione.

Sintassi
Int GetTotalCmpGates()

Parametri
-

Valore restituito
Un intero contenente il numero di porte composte definite nell'applicazione

Funzioni inerenti
GetTotalDigGates(),GetTotalEvnGates(),GetTotalNumGates(),GetTotalStrGates()

Esempio
```
TotalGates=GetTotalCmpGates();
```

9.7.4 StrGates

9.7.4.1 GetStrGateValue

Descrizione
Restituisce il valore della porta stringa specificata.

Sintassi
String GetStrGateValue(String Name, Int Id)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito
stringa della porta specificata

Funzioni inerenti
SetStrGateValue()

Esempio
```
CurrValue= GetStrGateValue("STRGate001",STRGateNo);
```

9.7.4.2 GetStrGateCommunicationStatus

Descrizione:
Ritorna lo stato di comunicazione con una data porta stringa.

Sintassi:
Bool GetStrGateCommunicationStatus(string Name,int Id)

Parametri:
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta non è in errore di comunicazione;
False= la porta non è stata trovata o è in errore di comunicazione

Funzioni inerenti:
GetStrGateValue(),SetStrGateValue()

Esempio:
```
...
if(GetStrGateCommunicationStatus(STR2,ID2)) then
   GetStrGateValue(STR2,ID2);
   ...
end
...
```

9.7.4.3 GetStrGateGateID

Descrizione
Restituisce il nome (campo "*ID Porta*" in *GateBuilder*) della porta stringa indicata.

Sintassi
String GetStrGateGateID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte stringa.

Valore restituito
GateID sotto forma di stringa

Funzioni inerenti
GetStrGateNID()

Esempio
```
string GateID;
GateID = GetStrGateGateID (10);
```

9.7.4.4 GetStrGateNID

Descrizione
Restituisce l'identificatore (campo "*N ID*" in *GateBuilder*) della porta stringa indicata.

Sintassi
Int GetStrGateNID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte stringa.

Valore restituito
Gate NID sotto forma di intero

Funzioni inerenti
GetStrGateGateID()

Esempio
```
int GateNID;
GateNID = GetStrGateNID (10);
```

9.7.4.5 GetStrGateProp

Descrizione
Restituisce una proprietà della porta stringa sotto forma di stringa.

Sintassi
String GetStrGateProp(String Name, Int Id, int Property)

Parametri
Name nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id identificatore della porta (campo "*N ID*" in *GateBuilder*)
Property numero della proprietà che si vuole ottenere

PROPRIETA'	COSTANTE NUMERICA
numero del canale	0

numero del dispositivo	1
indirizzo della porta	2
frequenza di campionamento	10
stato lettura porta	11
stato scrittura porta	12

Valore restituito
proprietà richiesta sotto forma di stringa

Funzioni inerenti
-
Esempio
```
samplingRate = GetStrGateProp (DIG1, ID1, 10); // frequenza di campionamento!
```

9.7.4.6 GetTotalStrGates

Descrizione
Restituisce il numero di porte stringa definite nell'applicazione.

Sintassi
Int GetTotalStrGates()

Parametri
-
Valore restituito
Un intero contenente il numero di porte stringa definite nell'applicazione

Funzioni inerenti
GetTotalDigGates(),GetTotalEvnGates(),GetTotalNumGates(),GetTotalCmpGates()

Esempio
```
TotalGates=GetTotalStrGates();
```

9.7.4.7 SetStrGateInMonitor

Descrizione
Abilita/disabilita il campionamento delle porte stringa definite da campionare "Se in schema".

Sintassi
Bool SetStrGateInMonitor(String Name, Int Id, Bool Enable)

Parametri

Name	nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id	identificatore della porta (campo "*N ID*" in *GateBuilder*)
Enable	"true" abilita il campionamento della porta , "false" disabilita il campionamento della porta.

Valore restituito
true (se la porta esiste)
false (in caso contrario)

Funzioni inerenti
SetNumGateInMonitor(),SetDigGateInMonitor()

Esempio
```
SetStrGateInMonitor(PrimaryName,PrimaryID,true);
```

9.7.4.8 SetStrGateValue

Descrizione
Modifica il valore della porta stringa specificata.

Sintassi
Bool SetStrGateValue(String Name, Int Id, String Value)

Parametri

Name	nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id	identificatore della porta (campo "*N ID*" in *GateBuilder*)
Value	nuovo valore da assegnare alla porta

Valore restituito
true (se la porta esiste ed è stata modificata)
false (in caso contrario)

Funzioni inerenti
GetStrGateValue()

Esempio
```
GateModified= SetStrGateValue("Gate0001SS",1,"Alfa");
```

9.7.4.9 StrGateExists

Descrizione:
Verifica se la porta stringa indicata è definita.

Sintassi:
Bool StrGateExists(string Name,int Id)

Parametri:

Name	nome della porta (campo "*ID Porta*" in *GateBuilder*)
Id	identificatore della porta (campo "*N ID*" in *GateBuilder*)

Valore restituito:
True= la porta è definita nell'applicazione
False= la porta non è definita nell'applicazione

Funzioni inerenti:
CmpGateExists() , DigGateExists() , EvnGateExists() , NumGateExists()

Esempio:
```
Bool ok;
ok = StrGateExists("S",1);
```

9.7.5 EvnGates

9.7.5.1 EvnGateExists

Descrizione:
Verifica se la porta evento indicata è definita.

Sintassi:
Bool EvnGateExists(string Name,int Id)

Parametri:

Name	nome della porta (campo "*Nome*" in *GateBuilder*)

Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito:
True= la porta è definita nell'applicazione
False= la porta non è definita nell'applicazione

Funzioni inerenti:
CmpGateExists() , DigGateExists() , NumGateExists() , StrGateExists()

Esempio:
```
Bool ok;
ok = EvnGateExists("E",1);
```

9.7.5.2 GetEvnGateAckedStatus

Descrizione:
Ritorna lo stato di acquisizione porta evento specificata.

Sintassi:
Bool GetEvnGateAckedStatus(string Name, int Id)

Parametri:
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito:
True= l'allarme è stato acquisito dall'utente.
False= l'allarme non è ancora stato acquisito dall'utente

Funzioni inerenti:
SetEvnGateAckedStatus()

Esempio
```
if(GetEvnGateAckedStatus(EVN3,ID3)==false) then
   GetEvnGateMsg(EVN3,ID3);
   ...
end
```

9.7.5.3 GetEvnGateExcludedStatus

Descrizione:
Ritorna lo stato di escluso della porta evento specificata.

Sintassi:
Bool GetEvnGateExcludedStatus(string Name,int Id)

Parametri:
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito:
True= l'allarme è escluso.
False= l'allarme NON è escluso.

Funzioni inerenti:
SetEvnGateExcludedStatus()

Esempio

```
if(GetEvnGateExcludedStatus(EVN3,ID3)==false) then
   GetEvnGateMsg(EVN3,ID3);
   ...
end
```

9.7.5.4 GetEvnGateGateID

Descrizione
Restituisce il nome della porta evento indicata (campo "*Nome*" in *GateBuilder*).

Sintassi
String GetEvnGateGateID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte evento.

Valore restituito
Stringa contenente il valore della porta.

Funzioni inerenti
GetEvnGateNID()

Esempio
string GateID;
GateID = GetEvnGateGateID (10);

9.7.5.5 GetEvnGateMsg

Descrizione:
Restituisce il messaggio associato alla porta evento specificata.

Sintassi:
String GetEvnGateMsg(String Name, Int Id)

Parametri:
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito:
messaggio associato alla porta

Funzioni inerenti:
GetEvnGateValue()

Esempio:
```
GateA_Msg = GetEvnGateMsg(GateA_Name,1);
```

9.7.5.6 GetEvnGateNID

Descrizione
Restituisce l'identificatore (campo "*ID*" in *GateBuilder*) della porta evento indicata.

Sintassi
Int GetEvnGateNID(Int Index)

Parametri
Index indice della porta nell'elenco delle porte evento.

Valore restituito
Numero intero contenente l'identificatore della porta

Funzioni inerenti
GetEvnGateGateID()

Esempio
```
int GateNID;
GateNID = GetEvnGateNID (10);
```

9.7.5.7 GetEvnGateValue

Descrizione
Restituisce il valore della porta evento specificata.

Sintassi
Bool GetEvnGateValue(String Name, Int Id)

Parametri
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito
- *true* : se l'evento è attivo
- *false*: se l'evento NON è attivo

Funzioni inerenti
GetEvnGateMsg()

Esempio
```
GateModified= GetEvnGateValue("Gate0001",1);
```

9.7.5.8 GetEvnGateSignificantStatus

Descrizione:
Ritorna lo stato "*significativo*" della porta evento specificata.
Un evento è considerato "*significativo*" se non è stato escluso e se:
- è attivo e non ancora acquisito (in caso di evento "non ritenuto")
- non più attivo ma non ancora acquisito (in caso di evento "ritenuto")

Sintassi:
Bool GetEvnGateSignificantStatus(string Name,int Id)

Parametri:
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito:
True= l'allarme è significativo.
False= l'allarme NON è significativo.

Esempio
```
if(GetEvnGateSignificantStatus(EVN3,ID3)==true) then
   GetEvnGateMsg(EVN3,ID3);
   ...
end
```

9.7.5.9 GetTotalEvnGates

Descrizione
Restituisce il numero di porte evento definite nell'applicazione.

Sintassi
Int GetTotalEvnGates()

Parametri
-

Valore restituito
Un intero contenente il numero di porte evento definite nell'applicazione

Funzioni inerenti
GetTotalDigGates(),GetTotalCmpGates(),GetTotalNumGates(),GetTotalStrGates()

Esempio
```
TotalGates=GetTotalEvnGates();
```

9.7.5.10 SetEvnGateAckedStatus

Descrizione
Segnala come "acquisita" la porta evento specificata.

Sintassi
Int SetEvnGateAckedStatus(string Name,int Id)

Parametri
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)

Valore restituito
True= operazione eseguita
False= porta non trovata

Funzioni inerenti
GetEvnGateAckedStatus()

Esempio
```
SetEvnGateAckedStatus("GateA",1);
```

9.7.5.11 SetEvnGateExcludedStatus

Descrizione
Esclude / Include la porta evento specificata.

Sintassi
Int SetEvnGateExcludedStatus(string Name,int Id, bool Exclude)

Parametri
Name nome della porta (campo "*Nome*" in *GateBuilder*)
Id identificatore della porta (campo "*ID*" in *GateBuilder*)
Exclude True : indica che la porta deve essere esclusa dal controllo.
False: indica che la porta deve essere inclusa nel controllo.

Valore restituito
True= operazione eseguita

False= porta non trovata

Funzioni inerenti
GetEvnGateExcludedStatus()

Esempio
```
SetEvnGateExcludedStatus("GateA",1,true);
```

9.8 Generic

9.8.1 AppendUserChangesEntry

Descrizione
Aggiunge un record allo storico interventi operatore.

Sintassivo
Void AppendUserChangesEntry(String Code,String Message)

Parametri

Code	messaggio da inserire nella colonna "Codice"
Message	messaggio da inserire nella colonna "Messaggio"

Valore restituito
-

Funzioni inerenti
-

Esempio
```
AppendUserChangesEntry("Section1","Machine stopped by user");
```

9.8.2 Beep

Descrizione
Emette un segnale acustico.

Sintassi
Void Beep()

Parametri
-

Valore restituito
-

Funzioni inerenti
-

Esempio
```
Beep();
```

9.8.3 CloseKeyboard

Descrizione
Chiude la tastiera precedentemente aperta.

Sintassi

Void CloseKeyboard()

Parametri
-

Valore restituito
Nessuno

Funzioni inerenti:
Keyboard()

Esempio
```
CloseKeyboard();
```

9.8.4 CloseSession

Descrizione*:*
Chiude la sessione in uso del programma supervisore

Sintassi:
Void CloseSession()

Parametri:
-

Valore restituito:
(nessuno)

Funzioni inerenti:
-

Esempio:
```
CloseSession();
```

9.8.5 CloseWindow

Descrizione
Chiude la finestra dalla quale si chiama la funzione.

Sintassi
Void CloseWindow()

Parametri
-

Valore restituito
-

Funzioni inerenti
WindowIsOpen()

Esempio
```
CloseWindow();
```

9.8.6 EnableShutdown

Descrizione:
Abilita e/o disabilita l'arresto del PC

Sintassi:
Void EnableShutdown(bool enable)

Parametri:
enable abilitazione spegnimento
"true" abilita lo spegnimento
"false" disabilita lo spegnimento)

Valore restituito:
(nessuno)

Funzioni inerenti:
GetShutdownStatus()

Esempio:
```
...
if (GetUserName()== administrator) then
    EnableShutDown(true);
end
...
```

9.8.7 Exec

Descrizione
Esegue un programma

Sintassi
Int Exec(String ProgName)

Parametri
ProgName Nome del programma da eseguire

Valore restituito
un valore maggiore di 31 (in caso l'operazione venga eseguita con successo)

Funzioni inerenti
-

Esempio
```
if (Exec("c:\windows\command\command.com") > 31)  then
      Executed=true;
end
```

9.8.8 ExecEx

Descrizione
Esegue un programma

Sintassi
Int ExecEx(String ProgName,int WindowType)

Parametri
ProgName il nome del programma da eseguire
WindowTipe
Se 1 allora esegue il programma in una finestra massimizzata.
Se 2 allora esegue il programma in una finestra minimizzata.

Se 3 allora esegue il programma in una finestra con le dimensione di default associate al programma stesso.

Valore restituito
un valore maggiore di 31 (in caso l'operazione venga eseguita con successo)

Funzioni inerenti
-

Esempio
```
if (ExecEx("c:\windows\command\command.com",1) > 31)  then
      Executed=true;
end
```

9.8.9 FileOpenDialog

Descrizione
Visualizza il dialogo standard di apertura file.

Sintassi
String FileOpenDialog(
string Filename ,
string FilenameFilter,
int FilterIndex,
string InitialDir,
string DefaultExt,
bool HideReadOnly,
bool PathMustExist,
bool FileMustExist,
bool NoValidate,
bool NoChangeDir,
bool AllowMultiSelect,
bool CreatePrompt,
bool NoReadOnlyReturn,
bool NoTestFileCreate,
bool OverwritePrompt,
bool ShareAware,
bool ShowHelp)

Parametri

string FileName	nome e percorso dell'ultimo file selezionato.
string FileNameFilter	maschera file (filtri)
int FilterIndex	filtro selezionato di default all'apertura del dialogo.
string InitialDir	directory corrente all'apertura del dialogo.
string DefaultExt	default dell'estensione file .
bool HideReadOnly	elimina il checkbox "Apri in sola lettura".
bool PathMustExist	genera un messaggio di errore se l'utente specifica un nome di file con un percorso inesistente.
bool FileMustExist	genera un messaggio di errore se l'utente seleziona un file non esistente.

bool NoValidate	disabilita il controllo sui caratteri non validi nel nome del file.Permette la selezione di file il cui none contiene dei cratteri non validi.
bool NoChangeDir	dopo che l'utente ha premuto "OK" resetta la directory corrente ponenedola uguale a quella indicata prima della selezione.
bool AllowMultiSelect	permette la selezione di più file.
bool CreatePrompt	genera un messaggio di warning se l'utente seleziona un file non esistente,chiedendo se si desidera creare un nuovo file con il nome specificato.
bool NoReadOnlyReturn	genera un messaggio di errore se l'utente prova a selezionare un file in sola lettura.
bool NoTestFileCreate	disabilita controllo sulla protezione dei file di rete e inaccessibilità dei dischi di rete. Applicabile solamente quando l'utente tenta di salvare un file in una directory con attributo "crate-no-modify".
bool OverwritePrompt	genera un messaggio di warning quando l'utente prova a selezionare un file che risulta già in uso,chiedendo se si desidera sovrascrivere il file essitente.
bool ShareAware	ignora gli errori di condivisione e consente che un file sia selezionato anche in caso di errori di condivisione.
bool ShowHelp	visualizza il pulsante di Help.

Valore restituito
Una stringa contenente il percorso+ il nome del file selezionato se è stato premuto il pulsante "OK"
Una stringa contenente "" se è stato premuto il pulsante "ESC"

Funzioni inerenti
FileSaveDialog()

Esempio

```
   string FileName;

   FileName=FileOpenDialog("",//Filename .
                           "*.txt",//Filename filter & filter patterns.
                           1,//Fiflter Index
                           "C:\",//Initial dir.
                           "*",//Default extension
                           true,// HideReadOnly
                           false,// PathMustExist
                           false,// FileMustExist
                           false,// NoValidate
                           false,// NoChangeDir
                           false,// AllowMultiSelect
                           false,// CreatePrompt
                           false,// NoReadOnlyReturn
                           false,// NoTestFileCreate
                           false,// OverwritePrompt
                           false,// ShareAware
                           false); // ShowHelp
```

```
MessageBox(FileName,"File Selected");
```

9.8.10 FileSaveDialog

Descrizione
Visualizza il dialogo standard di salvataggio file.

Sintassi
String FileSaveDialog(
string Filename ,
string FilenameFilter,
int FilterIndex,
string InitialDir,
string DefaultExt,
bool HideReadOnly,
bool PathMustExist,
bool NoValidate,
bool NoChangeDir,
bool AllowMultiSelect,
bool CreatePrompt,
bool NoReadOnlyReturn,
bool NoTestFileCreate,
bool OverwritePrompt,
bool ShareAware,
bool ShowHelp)

Parametri

string FileName	nome e percorso dell'ultimo file selezionato.
string FileNameFilter	maschera file (filtri)
int FilterIndex	filtro selezionato di default all'apertura del dialogo.
string InitialDir	directory corrente all'apertura del dialogo.
string DefaultExt	default dell'estensione file .
bool HideReadOnly	elimina il checkbox "Apri in sola lettura".
bool PathMustExist	genera un messaggio di errore se l'utente specifica un nome di file con un percorso inesistente.
bool NoValidate	disabilita il controllo sui caratteri non validi nel nome del file.Permette la selezione di file il cui none contiene dei cratteri non validi.
bool NoChangeDir	dopo che l'utente ha premuto "OK" resetta la directory corrente ponenedola uguale a quella indicata prima della selezione.
bool AllowMultiSelect	permette la selezione di più file.
bool CreatePrompt	genera un messaggio di warning se l'utente seleziona un file non esistente,chiedendo se si desidera creare un nuovo file con il nome specificato.
bool NoReadOnlyReturn	genera un messaggio di errore se l'utente prova a selezionare un file in sola lettura.

bool NoTestFileCreate	disabilita controllo sulla protezione dei file di rete e inaccessibilità dei dischi di rete.Applicabile solamente quando l'utente tenta di salvare un file in una directory con attributo "crate-no-modify".
bool OverwritePrompt	genera un messaggio di warning quando l'utente prova a selezionare un file che risulta già in uso,chiedendo se si desidera sovrascrivere il file essitente.
bool ShareAware	ignora gli errori di condivisione e consente che un file sia selezionato anche in caso di errori di condivisione.
bool ShowHelp	visualizza il pulsante di Help.

Valore restituito
Una stringa contenente il percorso+ il nome del file selezionato se è stato premuto il pulsante "OK"
Una stringa contenente "" se è stato premuto il pulsante "ESC"

Funzioni inerenti
FileOpenDialog()

Esempio

```
string FileName;

FileName=FileSaveDialog("",//Filename .
                        "*.txt",//Filename filter & filter patterns.
                        1,//Filter Index
                        "C:\",//Initial dir.
                        "*",//Default extension
                        true,// HideReadOnly
                        false,// PathMustExist
                        false,// NoValidate
                        false,// NoChangeDir
                        false,// AllowMultiSelect
                        false,// CreatePrompt
                        false,// NoReadOnlyReturn
                        false,// NoTestFileCreate
                        false,// OverwritePrompt
                        false,// ShareAware
                        false);// ShowHelp

MessageBox(FileName,"File Selected");
```

9.8.11 GetProjectCaption

Descrizione
Restituisce il testo nella caption del progetto attuale

Sintassi
String GetProjectCaption()

Parametri
-

Valore restituito
una stringa contenente il testo della caption del progetto attuale

Funzioni inerenti
GetProjectPath()

Esempio
```
ProjName=GetProjectCaption();
```

9.8.12 GetProjectName

Descrizione
Restituisce il nome del progetto attuale

Sintassi
String GetProjectName()

Parametri
-

Valore restituito
una stringa contenente il nome del progetto attuale

Funzioni inerenti
GetProjectPath()

Esempio
```
ProjName=GetProjectName();
```

9.8.13 GetProjectPath

Descrizione
Restituisce la directory del progetto attuale

Sintassi
String GetProjectPath()

Parametri
-

Valore restituito
una stringa contenente la directory del progetto attuale

Funzioni inerenti
GetProjectName()

Esempio
```
ProjPath=GetProjectPath();
```

9.8.14 GetShutdownStatus

Descrizione:
Restituisce lo stato di abilitazione del arresto del PC

Sintassi:
Bool GetShutdownStatus()

Parametri:
-

Valore restituito:
true se l'arresto del sistema è abilitato;
false se l'arresto è disabilitato

Funzioni inerenti:
EnableShutdown()

Esempio:
```
...
if (GetShutdownStatus()) then
 MessageBox("È ora possibile spegnere",  "Shutdown sequence:");
end
...
```

9.8.15 HexStrToInt

Descrizione
Converte un valore esadecimale contenuto in una stringa nel rispettivo valore intero

Sintassi
Int HexStrToInt(String HexValue)

Parametri
HexValue stringa contenente il valore da convertire

Valore restituito
un intero contenente il valore esadecimale della stringa

Funzioni inerenti
IntToHexStr()

Esempio
```
int Numero;
...
Numero=HexStrToInt("0C");     //Numero=12
...
```

9.8.16 Keyboard

Descrizione
Visualizza la tastiera specificata dal nome.
In caso di applicazione multilingua, verrà automaticamente caricata la tastiera denominata (nome)+"_"+(lingua corrente). (per esempio : "TastieraAlfanumerica_Italiano")
La tastiera deve essere stata definita tramite lo strumento Keyboard Builder.
Questa funzione viene utilizzata nelle applicazioni operanti su computer touch screen.
Ecco di seguito alcuni esempi di tastiere che possono essere costruite tramite Keyboard Builder:

Sintassi
Void Keyboard(String Name)

Parametri
Name nome della tastiera da aprire

Valore restituito
Nessuno

Funzioni inerenti:
CloseKeyboard()

Esempio
```
Keyboard("NumericKeyboard");
```

9.8.17 IconMessageBox

Descrizione
Mostra sullo schermo una finestra contenente del testo ed il titolo specificato più una combinazione di icone e pulsanti.Attende conferma da parte dell'utente.

Sintassi
Int IconMessageBox(String Text, String Title, Int ButtonType, Int IconType, Int DefaultButton)

Parametri

Text	stringa contenente il testo nella finestra
Title	stringa contenente il titolo della finestra
ButtonType	intero identificativo del tipo di tasti contenuti nella finestra
IconType	intero identificativo dell' icona visualizzata nella finestra
DefaultButton	intero indicante il tasto selezionato al apertura della finestra

ButtonType:
1 finestra contiene tre pulsanti: Annulla, Riprova, Ignora
2 finestra contiene un pulsante: OK
3 finestra contiene due pulsanti: OK, Cancella
4 finestra contiene due pulsanti: Riprova, Cancella
5 finestra contiene due pulsanti: SI, No
6 finestra contiene tre pulsanti: Si, No,Cancella

IconType
1 visualizza l'icona con un punto esclamativo (iconexclamation)
2 visualizza l'icona con un punto esclamativo (iconewarning)
3 visualizza l'icona con una lettera *i* minuscola contenuta in un cerchio (iconinformation)
4 visualizza l'icona con una lettera *i* minuscola contenuta in un cerchio (iconasterisk)
5 visualizza l'icona con un punto interrogativo (iconquestion)
6 visualizza l'icona con una *X* contenuta in un cerchio rosso (iconstop)
7 visualizza l'icona con una *X* contenuta in un cerchio rosso (iconerror)

8 visualizza l'icona con una *X* contenuta in un cerchio rosso (iconhand)

DefaultButton
1 il primo tasto è il tasto di default
2 il secondo tasto è il tasto di default
3 il terzo tasto è il tasto di default
4 il quarto tasto è il tasto di default

Valore restituito
0 se non è disponibile memoria per creare una nuova finestra
1 se è stato premuto il tasto Annulla
2 se è stato premuto il tasto Cancella
3 se è stato premuto il tasto Ignora
4 se è stato premuto il tasto No
5 se è stato premuto il tasto OK
6 se è stato premuto il tasto Riprova
7 se è stato premuto il tasto Si

Funzioni inerenti
QuestionBox(), InputDialog(), MessageBox

Esempio
```
IconMessageBox("È ora possibile spegnere",Shutdown sequence:", 2, 3,1);
```

9.8.18 InputDialog

Descrizione
Mostra sullo schermo una finestra di dialogo contenente un testo, un titolo specificato, un campo stringa modificabile e dei bottoni per la conferma o l'annullamento; per campo stringa si intende una stringa che l'utente può cambiare a suo piacimento tramite la tastiera.

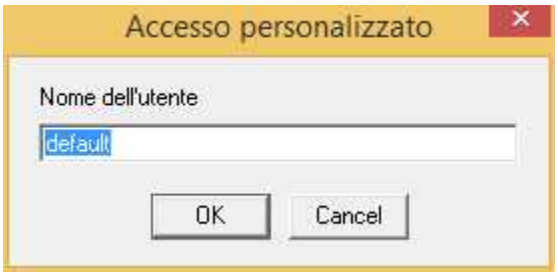
Sintassi
String InputDialog(String Text, String Title, String InitText)

Parametri
Text il testo contenuto nella finestra
Title il titolo della finestra
InitText il testo che inizialmente appare nel campo stringa

Valore restituito
il campo stringa nel caso venga data la conferma altrimenti una stringa nulla

Funzioni inerenti
MessageBox(), QuestionBox()

Esempio
```
User=InputDialog("Nome dell'utente", "Accesso personalizzato","default");
```



9.8.19 IntToHexStr

Descrizione
Converte un valore intero in esadecimale restituendolo sotto forma di stringa

Sintassi
String IntToHexStr(Int Value, Int Digits)

Parametri
Value valore intero da convertire
Digits specifica il minimo di cifre della stringa esadecimale, se il numero esadecimale fosse più corto allora verrebbero agiunti degli zeri a sinistra.

Valore restituito
stringa contenente la rappresentazione esadecimale del valore intero

Funzioni inerenti
HexStrToInt()

Esempio
```
Pattern=IntToHexStr(12800,4);
```

9.8.20 MessageBeep

Descrizione:
Emette un segnale acustico specificato.

Sintassi:
Void MessageBeep(int type)

Parametri:
type valore numerico associato al tipo di messaggio acustico desiderato

Valore	***Messaggio acustico***
1	IconAsterisk
2	IconExclamation
3	IconHand
4	IconQuestion
5	OK

(se il parametro type non assume nessun valore specificato sopra emette un tono singolo)

Valore restituito:
(nessuno)

Funzioni inerenti:
(nessuna)

Esempio:

```
MessageBeep(5);
```

9.8.21 MessageBox

Descrizione
Mostra sullo schermo una finestra contenente del testo ed il titolo specificato ed attende conferma da parte dell'utente.

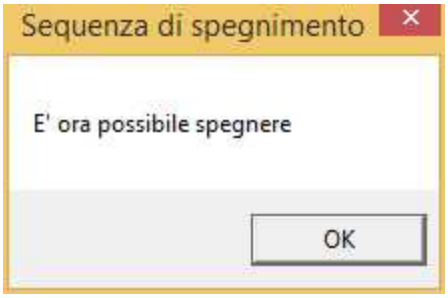
Sintassi
Void MessageBox(String Text, String Title)

Parametri
Text il testo contenuto nella finestra
Title il titolo della finestra

Valore restituito
-

Funzioni inerenti
QuestionBox(), InputDialog(), IconMessageBox()

Esempio
```
MessageBox("È ora possibile spegnere",Shutdown sequence:");
```



9.8.22 Play

Descrizione
Riproduce un file audio con estensione WAV.
Questa funzione è obsoleta.
Fare riferimento alla funzione PlaySound()

Sintassi
Void Play(String AudioFileName)

Parametri
AudioFileName il file da riprodurre

Valore restituito
-

Funzioni inerenti
PlaySound()

Esempio
```
Play("Z:\Audio\WAV\Alarm.wav");
```

9.8.23 PlaySound

Descrizione
Riproduce un file audio con estensione WAV o MP3.

Sintassi
```
Void  PlaySound(String AudioFileName,bool WaitMode,int RepeatCounter)
```

Parametri

AudioFileName	il file da riprodurre
WaitMode	se impostato a "true" allora la funzione attende fino alla conclusione della riproduzione del suono prima di passare all'istruzione successiva.
RepeatCounter	numero di volte che il suono deve essere ripetuto. Se è impostato a 0 allora il suono viene ripetuto continuamente e deve essere terminato tramite la funzione StopSound().

Valore restituito
-

Funzioni inerenti
StopSound()

Esempio
```
Play("Z:\Audio\WAV\Alarm.wav",false,0);
```

9.8.24 PrintString

Descrizione:
Stampa una stringa su un dispositivo

Sintassi:
Bool PrintString(String Msg, String Dev, Bool AddLF)

Parametri:

Msg	messaggio da stampare
Dev	dispositivo su cui stamparlo (LPT1, LPT2, etc)
AddLF	aggiungi il LineFeed

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso contrario)

Funzioni inerenti:
-

Esempio:
```
PrintString ("Inizializzazione fallita!","LPT1",True);
```

9.8.25 QuestionBox

Descrizione
Mostra sullo schermo una finestra contenente del testo ed il titolo specificato ed attente che l'utente effettui una celta binaria Si oppure No.

Sintassi
Bool QuestionBox(String Text, String Title)

Parametri

Text il testo contenuto nella finestra
Title il titolo della finestra

Valore restituito
true (nel caso l'utente scelga Si)
false (nel caso l'utente scelga No)

Funzioni inerenti
MessageBox(), InputDialog()

Esempio
```
AttivaMotori=QuestionBox("Attivo i motori?", "Stato motori");
```

9.8.26 RealToInt

Descrizione
Converte un valore reale in uno intero

Sintassi
int RealToInt(Real Value)

Parametri
Value valore reale da convertire

Valore restituito
il valore reale convertito in intero e quindi privato della parte decimale

Funzioni inerenti
-

Esempio
```
real TR = 123.45;
int Temp = RealToInt(TR);
```

9.8.27 ShellExec

Descrizione
La funzione ShellExec esegue una azione su un file. Il file può essere un eseguibile o un documento.

Sintassi
```
Int ShellExec(String FileName,String Verb,String WorkDir,int
              ShowMode,String Class,String Args)
```

Parametri
FileName specifica il nome del file da aprire o stampare. La funzione può aprire un file eseguibile o un documento, mentre può stampare un file documento. Se il percorso non è incluso nel nome allora viene utilizzata la directory corrente.

Verb specifica l'azione da eseguire. Il default è "Open" .

WorkDir specifica la directory di lavoro. Come default usa la directory corrente.

ShowMode può essere uno dei valori descritti di seguito. Se FileName indica un file eseguibile , allora ShowMode specifica come deve essere mostrata l'applicazione quando verrà eseguita. Se FileName indica un docuemnto allora ShowMode deve essere 0.
0 = for document file.
1 = SW_SHOW.
2 = SW_SHOWMAXIMIZED.
3 = SW_SHOWMINIMIZED.
4 = SW_HIDE.
5 = SW_MINIMIZE.
6 = SW_RESTORE.
7 = SW_SHOWMINNOACTIVE.
8 = SW_SHOWNA.
9 = SW_SHOWNOACTIVATE.
10=SW_SHOWNORMAL.

Class specifica il nome della classe del file o un identificatore GUID.

Args parametri aggiuntivi per l'applicazione. I parametri devono essee separati da spazio.

Valore restituito
un valore maggiore di 31 (in caso l'operazione venga eseguita con successo)

Funzioni inerenti
-

Esempio
```
ShellExec("MyDoc.pdf","open","C:\",0,".pdf","");
```

9.8.28 Sleep

Descrizione
Attende per un quanto di tempo specificato in millisecondi.

Sintassi
Void Sleep(int MilliSeconds)

Parametri
Milliseconds durata della pausa in millisecondi

Valore restituito
-

Funzioni inerenti
-

Esempio
```
Sleep(1000);
```

9.8.29 StopFunction

Descrizione:
Ferma una funzione in esecuzione

Sintassi:
Int StopFunction (String FunctionName)

Parametri:
FunctionName nome della funzione in esecuzione

Valore restituito:
0 in caso di successo
-1 funzione inesistente
-2 funzione non in esecuzione

Funzioni inerenti:
-

Esempio:
```
int res;
res=StopFunction("Reader");
...
...
Function void Reader()
...
end
```

9.8.30 StopSound

Descrizione
Termina l'eseguzione di un suono lanciata dall'istruzione PlaySound()

Sintassi
```
Void  StopSound()
```

Parametri:
-

Valore restituito
-

Funcioni inerenti:
PlaySound()

Esempio:
```
StopSound();
```

9.8.31 WindowIsOpen

Descrizione
Restituisce lo stato della finestra da cui si chiama la funzione.

Sintassi
Bool WindowIsOpen()

Parametri
-

Valore restituito
true (se la finestra (Es. un template), da cui è stata richiamata la funzione, è ancora aperta.)
false (in caso contrario)

Funzioni inerenti
CloseWindow()

Esempio
```
StatoFinestraMadre=WindowIsOpen();
```

9.9 Internet

9.9.1 SendMail

Descrizione
Invia una e-mail con un eventuale allegato.
Lavora su una connessione internet già attiva.

Sintassi
```
string SendMail(
                int Timeout,
                string HostSMTPServer,
                string UserName,
                string Password,
                string AddressFrom,
                string AddressTo,
                string AddressCc,
                string Subject,
                string Body,
                bool  AttachmentPresent,
                string AttachmentPath)
```

Parametri

Timeout	Timeout per il completamento dell'operazione (ms).
HostSMTPServer	Indirizzo server SMTP può: essere specificato nei seguenti modi: ***HostSMTPServer*** - connessione non crittografata su porta 25 ***HostSMTPServer:Porta*** - connessione non crittografata su porta specificata ***tls://HostSMTPServer*** - connessione crittografata TLS su porta 587 (default per TLS) ***tls://HostSMTPServer:Porta*** - connessione crittografata TLS su porta specificata ***ssl://HostSMTPServer*** - connessione crittografata SSL su porta 465 (default per SSL) ***ssl://HostSMTPServer:Porta*** - connessione crittografata SSL su porta specificata
UserName	Username usato per l'accesso al server SMTP.
UserPassword	Password usata per l'autenticazione al server SMTP.
AddressFrom	Indirizzo e-mail sorgente (N.B. Alcuni server SMTP bloccano e-mail provenienti da indirizzi inesistenti).
AddressTo	Indirizzo e-mail destinatario (N.B. è possibile inserire più indirizzi destinatario separati da virgola).
AddressCc	Indirzzo e-mail destinatario copia carbone (N.B. è possibile inserire più indirizzi destinatario separati da virgola).
Subject	Titolo dell'e-mail.
Body	Testo dell'e-mail.
AttachmentPresent	Se "***true***" invia e-mail con allegato.

AttachmentPath Path completo del file da allegare alla e-mail.

N.B. Se si utilizza un account GMAIL è necessario ABILITARE l'accesso per App meno sicure. Questa operazione può essere fatta sull'account tramite l'apposita pagina
https://www.google.com/settings/security/lesssecureapps
Maggiori informazioni all'indirizzo https://support.google.com/accounts/answer/6010255?hl=en

Valore di ritorno
Se non si sono verifcati errori allora viene restituita un stringa vuota, altrimenti viene restituito il testo del messaggio di errore.

Esempio

```
Function void SendMailToGmail()
//*************************************************************************
****
// Send Mail demo procedure
//*************************************************************************
****
      string Error = SendMail(10000,
                                          "tls://smtp.gmail.com",
                                          "my.address@gmail.com",
                                          "MyPassword",
                                          "my.address@gmail.com",
                                          "recipient_1@company.com,
recipient_2@company.com",
                                          "recipient_3@company.com",
                                          "Test e-mail",
                                          "This is a test email sent by
Winlog",
                                          false, "");

      if (Error == "") then
            MessageBox("Operation completed", "Send Mail");
      else
            MessageBox(Error, "Send Mail");
      end
end
```

9.9.2 FTP

9.9.2.1 FTPConnect

Descrizione
Apre una connessione FTP verso un server FTP remoto.

Sintassi

```
     int FTPConnect(
                        int FTPPort,
                        string FTPServerHost,
                        string UserName,
                        string Password,
                        bool Passive,
                        int Timeout)
```

Parametri

FTPPort	Porta TCP dal server FTP.
FTPServerHost	Indirizzo server FTP può essere specificato nei seguenti modi: **FTPServerHost -** Connessione con crittografia automatica, viene tentata la connessione TLS, altrimenti non si utilizza alcuna crittografia. **ftp://FTPServerHost -** Connessione non crittografata. **ftps://FTPServerHost -** Connessione con crittografia TLS.
UserName	Username usato per l'accesso al server FTP.
UserPassword	Password usata per l'autenticazione al server FTP.
Passive	Metodo di connessione al server FTP: **true** = metodo "Passivo", **false** = metodo "Attivo".
Timeout	Timeout per il completamento dell'operazione (ms).

Valore di ritorno

0 - 9	handle della connessione, connessione avvenuta correttamente.
-1	indica che non ci sono connessioni disponibili (possono essere aperte fino ad un massimo di 10 connessioni contemporanee).
-2	indica errore generico.

Funzioni inerenti

FTPDisconnect(), FTPGet(), FTPPut(), FTPDelete(), FTPMakeDir(), FTPRemoveDir()

Esempio

```
Function void Connect()
//*************************************************************************
****
// FTP Connection
//*************************************************************************
****
#modal
    int Handle;
    Handle = FTPConnect(21, "ftp.test.com", "myusername", "mypassword",
false, 10000);
    if (Handle == -1) then
        MessageBox("No available connections", "Error");
    end

    if (Handle == -2) then
        MessageBox("Generic error", "Error");
    end

    SetNumGateValue("Connection", 1, Handle);
end
```

9.9.2.2 FTPDelete

Descrizione

Elimina un file da una server FTP remoto.

Sintassi

```
Bool FTPDelete(
                  int Handle,
                  string FileName)
```

Parametri

Handle	Handle della connessione precedentemente aperta tramite la funzione FTPConnect().
FileName	Nome del file da eliminare sul server FTP (percorso completo).

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPDisconnect(), FTPGet(), FTPPut(), FTPMakeDir(), FTPRemoveDir()

Esempio

```
Function void Delete()
//*************************************************************************
****
// FTP Delete
//*************************************************************************
****
#modal
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPDelete(Handle, "myFtpRemoteFolder/myFile.txt");

    if (Ret == false) then
        MessageBox("Error on FTP DELETE", "Error");
    else
        MessageBox("DELETE operation completed successfully", "Info");
    end
end
```

9.9.2.3 FTPDisconnect

Descrizione
Chidue una connessione FTP precedentemente aperta verso un server FTP.

Sintassi

```
Bool FTPDisconnect(int Handle)
```

Parametri
Handle Handle della connessione precedentemente aperta tramite la funzione FTPConnect()

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPGet(), FTPPut(), FTPDelete(), FTPMakeDir(), FTPRemoveDir()

Esempio

```
Function void Disconnect()
```

```
//*************************************************************************
****
// FTP Disconnection
//*************************************************************************
****
#modal
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPDisconnect(Handle);

    if (Ret == false) then
        MessageBox("Disconnection error!", "Error");
    else
        MessageBox("Disconnection Ok!", "Info");
    end
end
```

9.9.2.4 FTPGet

Descrizione
Scarica un file da un server FTP tramite il protocollo FTP.

Sintassi

```
Bool FTPGet(
            int Handle,
            string SourceFileName,
            string DestinationFileName,
            int TransferType)
```

Parametri

Handle	Handle della connessione precedentemente aperta tramite la funzione FTPConnect().
SourceFileName	Nome del file sorgente presente sul server FTP (percorso completo).
DestinationFileName	Nome del file destinazione che verrà creato sul PC locale (con percorso completo).
TransferType	Se 0 il trasferimento del file viene effettuato in modalità Binaria, mentre se è 1 viene trasferito in modalità ASCII..

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPDisconnect(), FTPPut(), FTPDelete(), FTPMakeDir(), FTPRemoveDir()

Esempio

```
Function void Get()
//*************************************************************************
****
// FTP Get
//*************************************************************************
****
#modal
```

```
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPGet(Handle, "myFtpRemoteFolder/myFile.txt",
                         GetProjectPath() + "\myFile.txt",
                         0);

    if (Ret == false) then
        MessageBox("Error on FTP GET", "Error");
    else
        MessageBox("GET operation completed successfully", "Info");
    end
end
```

9.9.2.5 FTPMakeDir

Descrizione
Crea una directory su un server FTP remoto.

Sintassi
```
Bool FTPMakeDir(
             int Handle,
             string DirectoryName)
```

Parametri

Handle	Handle della connessione precedentemente aperta tramite la funzione FTPConnect().
DirectoryName	Nome della directory da creare sul server FTP (percorso completo).

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPDisconnect(), FTPGet(), FTPPut(), FTPDelete(), FTPRemoveDir()

Esempio
```
Function void MakeDir()
//*************************************************************************
****
// FTP MakeDir
//*************************************************************************
****
#modal
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPMakeDir(Handle, "myFtpRemoteFolder/myDirectory/");

    if (Ret == false) then
        MessageBox("Error on FTP MAKE DIR", "Error");
    else
        MessageBox("MAKE DIR operation completed successfully", "Info");
    end
```

```
end
```

9.9.2.6 FTPPut

Descrizione
Carica un file su un server FTP remoto usando il protocollo FTP.

Sintassi
```
Bool FTPPut(
             int Handle,
             string SourceFileName,
             string DestinationFileName,
             int TransferType)
```

Parametri

Handle	Handle della connessione precedentemente aperta tramite la funzione FTPConnect().
SourceFileName	Nome del file sorgente presente sul PC locale (percorso completo).
DestinationFileName	Nome del file destinazione che verrà creato sul server FTP (con percorso completo).
TransferType	Se 0 il trasferimento del file viene effettuato in modalità Binaria, mentre se è 1 viene trasferito in modalità ASCII.

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPDisconnect(), FTPGet(), FTPDelete(), FTPMakeDir(), FTPRemoveDir()

Esempio
```
Function void Put()
//*************************************************************************
****
// FTP Put
//*************************************************************************
****
#modal
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPPut(Handle, GetProjectPath() + "\myFile.txt",
                         "myFtpRemoteFolder/myFile.txt",
                         0);

    if (Ret == false) then
        MessageBox("Error on FTP PUT", "Error");
    else
        MessageBox("PUT operation completed successfully", "Info");
    end
end
```

9.9.2.7 FTPRemoveDir

Descrizione
Elimina una directory da un server FTP remoto.

Sintassi
```
Bool FTPRemoveDir(
                  int Handle,
                  string DirectoryName,
                  bool DeleteIfNotEmpty)
```

Parametri

Handle	Handle della connessione precedentemente aperta tramite la funzione FTPConnect().
DirectoryName	Nome della directory da Eliminare sul server FTP (percorso completo).
DeleteIfNotEmpty	Se **true** elimina la directory anche se non è vuota, se **false** elimina la directory solo se è vuota.

Valore di ritorno
True: operazione completata con successo.
False: errore

Funzioni inerenti
FTPConnect(), FTPDisconnect(), FTPGet(), FTPPut(), FTPDelete(), FTPMakeDir()

Esempio
```
Function void RemoveDir()
//*************************************************************************
****
// FTP RemoveDir
//*************************************************************************
****
#modal
    bool Ret;
    int Handle;
    Handle = GetNumGateValue("Connection", 1);
    Ret = FTPRemoveDir(Handle, "myFtpRemoteFolder/myDirectory/", true);
    if (Ret == false) then
        MessageBox("Error on FTP REMOVE DIR", "Error");
    else
        MessageBox("REMOVE DIR operation completed successfully", "Info");
    end
end
```

9.10 Math

9.10.1 Abs

Descrizione
Restituisce il valore assoluto dell'argomento passato.

Sintassi
Real Abs(Real Value)

Parametri
Value numero di cui effettuare il valore assoluto

Valore restituito
il valore assoluto dell'argomento

Funzioni inerenti
-

Esempio
```
PosValue=Abs(NegValue);
```

9.10.2 ArcCos

Descrizione
Calcola l'arcocoseno di un valore reale compreso tra -1 e 1

Sintassi
Real ArcCos(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
l'arcocoseno del valore specificato

Funzioni inerenti
Cos(), ArcSin()

Esempio
```
newval=ArcCos(value);
```

9.10.3 ArcTan

Descrizione
Calcola l'arcotangente di un valore reale

Sintassi
Real ArcTan(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
l'arcotangente del valore specificato

Funzioni inerenti
Tan()

Esempio
```
Theta=ArcTan(Y/X);
```

9.10.4 ArcSin

Descrizione
Calcola l'arcoseno di un valore reale compreso tra -1 e 1

Sintassi
Real ArcSin(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
l'arcoseno del valore specificato

Funzioni inerenti
ArcCos(), Sin()

Esempio
```
newval=ArcSin(value);
```

9.10.5 Cos

Descrizione
Calcola il coseno di un valore reale (gli angoli sono in radianti)

Sintassi
Real Cos(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
il coseno del valore specificato

Funzioni inerenti
Sin(), ArcCos()

Esempio
```
SinTheta=Sin(Theta);
```

9.10.6 Exp

Descrizione
Calcola l'esponenziale di un valore reale

Sintassi
Real Exp(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
l'esponenziale del valore specificato

Funzioni inerenti

Log()

Esempio
```
Delta=Exp(2.758);
```

9.10.7 Log

Descrizione
Calcola il logartimo naturale di un valore reale

Sintassi
Real Log(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
il logaritmo del valore specificato

Funzioni inerenti
Exp()

Esempio
```
SimilZero=Log(1.00001);
```

9.10.8 Mod

Descrizione
Restituisce il resto di una divisione tra interi (A / B)

Sintassi
Int Mod(int A, int B)

Parametri
A dividendo
B divisore

Valore restituito
il modulo della divisione

Funzioni inerenti
-

Esempio
```
Resto=Mod(44,6);
```

9.10.9 Pow

Descrizione
Calcola l'elevamento a potenza di un valore reale

Sintassi
Real Pow(Real Base, Real Exponent)

Parametri

Base la base, il valore di cui effettuare l'elevamento
Exponent l'esponente

Valore restituito
l'elevamento a potenza calcolato

Funzioni inerenti
Sqrt()

Esempio
```
PowerOfTwo=Pow(2,8);
```

9.10.10 Rand

Descrizione
Restituisce un valore casuale scelto in un range specificato

Sintassi
Int Rand(Int Range)

Parametri
Range dimensione del dominio di scelta

Valore restituito
un valore casuale compreso tra 0 e (Range-1)

Funzioni inerenti
-

Esempio
```
LottoNo=Rand(90)+1;
```

9.10.11 Round

Descrizione
Arrotonda un numero al decimale specificato.

Sintassi
Real Round(Real Value, Int Decimals)

Parametri
Value numero da arrotondare
Decimals decimale a cui effettuare l'arrtondamento

Valore restituito
il numero arrotondato

Funzioni inerenti
-

Esempio
```
LastValue=Round(Sampled,3);
```

9.10.12 Sin

Descrizione
Calcola il seno di un valore reale (gli angoli sono in radianti)

Sintassi
Real Sin(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
il seno del valore specificato

Funzioni inerenti
Cos(), ArcSin()

Esempio
```
SinTheta=Sin(Theta);
```

9.10.13 Sqrt

Descrizione
Calcola la radice quadrata di un valore reale

Sintassi
Real Sqrt(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
la radice quadrata del dato specificato

Funzioni inerenti
Pow()

Esempio
```
Val=Sqrt(A*B);
```

9.10.14 Tan

Descrizione
Calcola la tangente di un valore reale (gli angoli sono in radianti)

Sintassi
Real Tan(Real Value)

Parametri
Value valore di cui effettuare il calcolo

Valore restituito
la tangente del valore specificato

Funzioni inerenti

ArcTan()

Esempio
```
TanAlfa=Tan(Beta-Gamma);
```

9.11 Modem

9.11.1 ModemAvailable

(OBSOLETA)
Si consiglia di utilizzare la funzione SMSOpenChannel().

Descrizione
Verifica che sulla porta seriale specificata sia collegato e pronto il modem GSM

Sintassi
int ModemAvailable(int SerialPortNumber)

Parametri
SerialPortNumber indical il numero della porta seriale:
1 - indica la porta COM1
2 - indica la porta COM2
x - indica la generica porta COMx

Valore restituito
il risultato dell'operazione:
0 Il modem GSM risponde ed è pronto
1 (interno) non è possibile creare il thread d'ascolto
2 (interno) non è possibile allocare memoria per i buffers
-1 impossibile aprire la porta seriale (COM) specificata
-2 impossibile aprire la porta (porta già aperta)
-3 il modem ha risposto ERRORE (è possibile che non sia un modem GSM)
-4 il modem ha risposto in maniera non standard (risposta diversa da OK o ERROR)
-5 il modem non risponde

Funzioni inerenti
ModemSetPIN() , ModemSetServicesCenter() , ModemSetTextMode() , ModemSendSMS()

Esempio
```
if (ModemAvailable(2) !=0) then
      MessageBox("Il modem sulla COM2 non è pronto","MODEM");
end
```

9.11.2 ModemDial

Descrizione
Chiama un dispositivo remoto ad un numero telefonico specificato.
(Il sistema deve avere un modem installato)

Sintassi
Int ModemDial(int Channel, String Number, int TimeOut)

Parametri
Channel il numero del canale su cui avverrà la comunicazione (su questo canale dovrà essere selezionato un protocollo remoto)

Number	il numero telefonico del dispositivo da chiamare
TimeOut	un tempo massimo entro il quale va stabilità la connessione

Valore restituito
il risultato dell'operazione:
0 Connesso
-1 Init Failed (inizializzaione funzioni telefoniche fallita)
-2 Dial failed (chiamata telefonica fallita)
-3 Busy (linea occupata, ecc)
-4 TimeOut (tempo di connessione esaurito)
-5 Modem already in use (il modem sta già comunincando)
-6 Channel is not remote (il canale selezionato non possiede un protocollo remoto)
-7 Ghost Channel (il canale specificato non possiede un protocollo selezionato)

Funzioni inerenti
ModemHangup()

Esempio
```
int res;
res=ModemDial(1,"800600600",30);
```

9.11.3 ModemHangup

Descrizione
Chiude la connesione in corso con il dispositivo remoto

Sintassi
void ModemHangup()

Parametri
-

Valore restituito
-

Funzioni inerenti
ModemDial()

Esempio
```
ModemHangup();
```

9.11.4 ModemSendSMS

(OBSOLETA)
Si consiglia di utilizzare la funzione SMSSend().

Descrizione
Spedisce un SMS testuale al numero specificato

Sintassi
int ModemSendSMS(int SerialPortNumber, string Number, string Message)

Parametri

SerialPortNumber	indical il numero della porta seriale: 1 - indica la porta COM1 2 - indica la porta COM2 x - indica la generica porta COMx

Number	il destinatario per il messaggio (tutti gli apparecchi abilitati alla ricezione degli SMS)
Message	il testo del messaggio da spedire

Valore restituito
il risultato dell'operazione:
0 Il modem GSM risponde ed è pronto
1 (interno) non è possibile creare il thread d'ascolto
2 (interno) non è possibile allocare memoria per i buffers
-1 impossibile aprire la porta seriale (COM) specificata
-2 impossibile aprire la porta (porta già aperta)
-3 il modem ha risposto ERRORE (è possibile che non sia un modem GSM)
-4 il modem ha risposto in maniera non standard (risposta diversa da OK o ERROR)
-5 il modem non risponde
-6 il modem non ha riconosciuto il comando
-11 Il numero inserito è troppo lungo (massimo 40 caratteri)
-12 Il messaggio inserito è troppo lungo (massimo 160 caratteri)

Funzioni inerenti
ModemAvailable() , ModemSetPIN() , ModemSetServicesCenter() , ModemSetTextMode()

Esempio
```
string message;
message = "Allarme di temperatura alta " + GetNumGateValue("temp",11)+ "
°C; intervenire";
ModemSendSMS(1, "+393491212121", message);
```

9.11.5 ModemSetPIN

(OBSOLETA)
Si consiglia di utilizzare la funzione SMSOpenChannel().

Descrizione
Imposta il PIN per accedere alla scheda telefonica (SIM) inserita nel modem GSM

Sintassi
int ModemSetPIN(int SerialPortNumber, string PIN)

Parametri

SerialPortNumber	indical il numero della porta seriale: 1 - indica la porta COM1 2 - indica la porta COM2 x - indica la generica porta COMx
PIN	il numero PIN per l'accesso alla SIM

Valore restituito
il risultato dell'operazione:
0 Il modem GSM risponde ed è pronto
1 (interno) non è possibile creare il thread d'ascolto
2 (interno) non è possibile allocare memoria per i buffers
-1 impossibile aprire la porta seriale (COM) specificata
-2 impossibile aprire la porta (porta già aperta)
-3 PIN errato (attenzione il numero di tentativi è limitato)
-4 il modem ha risposto in maniera non standard (risposta diversa da OK o ERROR)
-5 il modem non risponde

Funzioni inerenti
ModemAvailable() , ModemSetServicesCenter() , ModemSetTextMode() , ModemSendSMS()

Esempio
```
ModemSetPIN(2,"1234"); // imposta il codice PIN 1234 per accedere alla SIM
```

N.B.: Si consiglia di attendere circa 30 secondi prima di utilizzare di nuovo il modem GSM altrimenti l'autenticazione potrebbe non essere eseguita correttamente.

9.11.6 ModemSetServicesCenter

(OBSOLETA)
Si consiglia di utilizzare la funzione SMSOpenChannel().

Descrizione
Imposta il numero del centro servizi per poter spedire gli SMS, questa impostazione è necessaria.

Sintassi
int ModemSetServicesCenter(int SerialPortNumber, string ServicesCenterNumber)

Parametri

SerialPortNumber	indical il numero della porta seriale: 1 - indica la porta COM1 2 - indica la porta COM2 x - indica la generica porta COMx
ServicesCenterNumber	il numero del centro servizi (vedi nota in fondo alla pagina)

Valore restituito
il risultato dell'operazione:

0	Il modem GSM risponde ed è pronto
1	(interno) non è possibile creare il thread d'ascolto
2	(interno) non è possibile allocare memoria per i buffers
-1	impossibile aprire la porta seriale (COM) specificata
-2	impossibile aprire la porta (porta già aperta)
-3	il modem ha risposto ERRORE (è possibile che non sia un modem GSM)
-4	il modem ha risposto in maniera non standard (risposta diversa da OK o ERROR)
-5	il modem non risponde
-11	Il numero inserito è troppo lungo (massimo 40 caratteri)

Funzioni inerenti
ModemAvailable() , ModemSetPIN() , ModemSetTextMode() , ModemSendSMS()

Esempio
```
ModemSetServicesNumber(2,"+393492000300");
```

N.B.: Utilizzando un cellulare è possibile leggere il numero del centro servizi navigando tra le impostazioni.
Lo stesso numero può essere richiesto al proprio gestore di telefonia o consultando il manuale allegato alla propria SIM.

9.11.7 ModemSetTextMode

(OBSOLETA)
Si consiglia di utilizzare la funzione SMSOpenChannel().

Descrizione
Imposta gli SMS nel formato testo

Sintassi
int ModemSetTextMode(int SerialPortNumber)

Parametri
SerialPortNumber indical il numero della porta seriale:
1 - indica la porta COM1
2 - indica la porta COM2
x - indica la generica porta COMx

Valore restituito
il risultato dell'operazione:
0 Il modem GSM risponde ed è pronto
1 (interno) non è possibile creare il thread d'ascolto
2 (interno) non è possibile allocare memoria per i buffers
-1 impossibile aprire la porta seriale (COM) specificata
-2 impossibile aprire la porta (porta già aperta)
-3 il modem ha risposto ERRORE
-3 il modem ha risposto ERRORE (è possibile che non sia un modem GSM)
-4 il modem ha risposto in maniera non standard (risposta diversa da OK o ERROR)
-5 il modem non risponde

Funzioni inerenti
ModemAvailable() , ModemSetPIN() , ModemSetServicesCenter() , ModemSendSMS()

Esempio
```
ModemSetPIN(2,"4488");
ModemSetServicesNumber(2,"+393492000300");
ModemSetTextMode(2);
```

9.12 Multilanguage

9.12.1 GetCurrentLanguage

Descrizione
Restituisce il linguaggio attualmente selezionato nell'applicazione.

Sintassi
String GetCurrentLanguage()

Parametri
-

Valore restituito
una stringa contenente il nome del linguaggio attualmente selezionato

Funzioni inerenti:
SetCurrentLanguage(),GetNextLanguage()

Esempio
```
string ProjLanguage;
ProjLanguage=GetCurrentLanguage();
```

9.12.2 GetNextLanguage

Descrizione
Ritorna la lingua successiva a quella indicata in ingresso.

Sintassi
```
string  GetNextLanguage(string Language)
```

Parametri
Language è il nome della lingua dalla quale deve essere cercata la successiva nell'elenco delle lingue definite per l'applicazione.
se viene specificata una stringa vuota ("") allora verrà ritornata la prima lingua dell'elenco.

Valore restituito
Il nome della lingua successiva o una stringa vuota se è stata raggiunta la fine dell'elenco.

Funzioni inerenti:
SetCurrentLanguage(),GetCurrentLanguage()

Esempio
```
string Language="";
Do
      Language=GetNextLanguage(Language);
      MessageBox(Language,"Language found");
while(Language!="")
```

9.12.3 SetCurrentLanguage

Descrizione
Imposta nell'applicazione la nuova lingua corrente.

Sintassi
```
bool  SetCurrentLanguage(string Language,bool Mode)
```

Parametri
Language nome della lingua da impostare.
Mode se "true" allora l'impostazione sarà permanente.
se "false" allora l'impostazione sarà temporanea (al successivo riavvio dell'applicazione l'impostazione verrà resettata).

Valore restituito
true (se operazione eseguita con successo)
false (in caso di fallimento)

Funzioni inerenti:
GetCurrentLanguage(),GetNextLanguage()

Esempio
```
SetCurrentLanguage("English",true);
```

9.13 Password

9.13.1 AddUser

Descrizione:
Permette di definire un nuovo utente con la relativa password e gruppo di appartenenza.

Sintassi:
```
Int AddUser(string UserName, string Password,int Groups,bool Overwrite)
```

Parametri:

UserName	utente da creare
Passoword	password associata all'utente
Groups	gruppo associato all'utente (Consultare GetUserGroups() per ulteriori dettagli)
Overwrite	"true" se l'utente deve essere sovrascritto nel caso in cui risultasse già definito.

Valore restiruito:

0	Utente aggiunto con successo
1	Lunghezza nome utente è 0
2	Lunghezza nome utente è superiore al massimo consentito
3	Lunghezza Password è 0 0
4	Lunghezza Password è superiore al consentito
5	Nessun gruppo associato
6	All'utente corrente non è consentito creare nuovi utenti.
7	Utente già defintio.
8	Errore di lettura/scrittura file delle password

Funzioni inerenti:
RemoveUser()

Esempio:

```
AddUser(GetStrGateValue("User",0),GetStrGateValue("Password",0),GetNumGateV
alue("Groups",0),true);
```

9.13.2 GetUserName

Descrizione
Restituisce il nome del utente attualmente collegato al programma di supervisone

Sintassi
string GetUserName()

Parametri
-

Valore restituito
una stringa contenente il nome del utente

Funzioni inerenti
GetProjectPath(),GetProjectName()

Esempio
```
UtenteAttuale=GetUserName();
```

9.13.3 GetUserGroups

Descrizione
Restituisce i gruppi di appartenenza dell'utente attualmente collegato al programma di supervisone

Sintassi
integer GetUserGroups()

Parametri
-

Valore restituito
un valore intero che rappresenta in forma binaria i gruppi di appartenenza dell'utente attuale.
Se per esempio l'utente attuale è associato al gruppo 1, al gruppo 3 ed al gruppo 7 il valore restituito da questa funzione sarà 69 cioè :

[2^00	* 1]	+	(Gruppo 1)	1+
[2^01	* 0]	+	(Gruppo 2)	0+
[2^02	* 1]	+	(Gruppo 3)	4+
[2^03	* 0]	+	(Gruppo 4)	0+
[2^04	* 0]	+	(Gruppo 5)	0+
[2^05	* 0]	+	(Gruppo 6)	0+
[2^06	* 1]	+	(Gruppo 7)	64+
[2^07	* 0]	+	(Gruppo 8)	0+
[2^08	* 0]	+	(Gruppo 9)	0+
[2^09	* 0]	+	(Gruppo 10)	0+
[2^10	* 0]	+	(Gruppo 11)	0+
[2^11	* 0]	+	(Gruppo 12)	0+
[2^12	* 0]	+	(Gruppo 13)	0+
[2^13	* 0]	+	(Gruppo 14)	0+
[2^14	* 0]	+	(Gruppo 15)	0

Funzioni inerenti
GetUserName()
Esempio
```
Gruppi=GetUserGroups();
```

9.13.4 Login

Descrizione:
Effettua l'accesso con l'utente specificato

Sintassi:
Void Login(string User,string Password)

Parametri:

User	utente con cui effettuare il login
Password	password associata all'utente

Valore restituito:
(nessuno)

Funzioni inerenti:
Logout()

Esempio
```
Login("User1","Operator");
```

9.13.5 Logout

Descrizione:
Scollega dal supervisore l'utente attualmente collegato
(pone automaticamente il valore UserName a None)

Sintassi:
Void Logout()

Parametri:
-

Valore restituito:
(nessuno)

Funzioni inerenti:
Login()

Esempio
```
Logout();
```

9.13.6 RemoveUser

Descrizione:
Rimuove dall'elenco degli utenti l'utente specificato.

Sintassi:
```
Int RemoveUser(string UserName)
```

Parametri:
UserName utente da rimuovere

Valore restiruito:
0 Utente rimosso con successo
1 Lunghezza nome utente è 0
2 Lunghezza nome utente è superiore al massimo consentito
6 All'utente corrente non è consentito eliminare altri utenti.
8 File delle password non trovato
9 Utente non trovato

Funzioni inerenti:
AddUser()

Esempio
```
RemoveUser(GetStrGateValue("User",0));
```

9.13.7 UserFindFirst

Descrizione
Inizia la lettura del file degli utenti

Sintassi
int UserFindFirst()

Parametri
-

Valore restituito
Handle asseganto al gestore dell'operazione di ricerca in corso. Questo handle deve essere successivamente usato anche nelle istruzioni UserFindNext() e UserFindClose().
Per sapere se è stato trovato almeno un utente controllare che UserNameFound() restituisca una stringa diversa da "".

Funzioni inerenti
UserFindNext(),UserFindClose(), UserNameFound(), UserGroupsFound()

Esempio
```
Function void FindUsers()
    int P=UserFindFirst();
    bool Found;
    if UserNameFound(P)!="" then Found=true; else Found=false;end
    While(Found==true)
        MessageBox(UserNameFound(P),UserGroupsFound(P));
        Found=UserFindNext(P);
    end
    UserFindClose(P);
end
```

9.13.8 UserFindNext

Descrizione
Continua la lettura dell'elenco degli utenti

Sintassi
bool UserFindNext(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
true (se è stato letto un altro utente)
false (non ci sono più utenti da leggere)

Funzioni inerenti
UserFindFirst(), UserFindClose(),UserNameFound(), UserGroupsFound()

Esempio
```
Function void FindUsers()
    int P=UserFindFirst();
    bool Found;
    if UserNameFound(P)!="" then Found=true; else Found=false;end
    While(Found==true)
        MessageBox(UserNameFound(P),UserGroupsFound(P));
        Found=UserFindNext(P);
    end
    UserFindClose(P);
end
```

9.13.9 UserFindClose

Descrizione
Termina l'operazione di lettura iniziata con User*FindFirst* e libera le risorse allocate a tale scopo

Sintassi

void UserFindClose(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
-

Funzioni inerenti
UserFindFirst(), UserFindNext(),UserNameFound(), UserGroupsFound()

Esempio
```
Function void FindUsers()
    int P=UserFindFirst();
    bool Found;
    if UserNameFound(P)!="" then Found=true; else Found=false;end
    While(Found==true)
        MessageBox(UserNameFound(P),UserGroupsFound(P));
        Found=UserFindNext(P);
    end
    UserFindClose(P);
end
```

9.13.10 UserGroupsFound

Descrizione
Restituisce i gruppi di appartenenza dell'utente letto con User*FindFirst* o User*FindNext*

Sintassi
String UserGroupsFound(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
Valore intero che rappresenta in forma binaria i gruppi di appartenenza dell'utente (Consultare GetUserGroups() per ulteriori dettagli).

Funzioni inerenti
UserFindNext(), UserNameFound(), UserGroupsFound(), UserFindClose()

Esempio
```
Function void FindUsers()
    int P=UserFindFirst();
    bool Found;
    if UserNameFound(P)!="" then Found=true; else Found=false;end
    While(Found==true)
        MessageBox(UserNameFound(P),UserGroupsFound(P));
        Found=UserFindNext(P);
    end
    UserFindClose(P);
end
```

9.13.11 UserNameFound

Descrizione
Restituisce il nome dell'utente letto con User*FindFirst* o User*FindNext*

Sintassi
String UserNameFound(int Handle)

Parametri
Handle gestore dell'operazione di ricerca in corso.

Valore restituito
il nome dell'utente letto

Funzioni inerenti
UserFindNext(), UserNameFound(), UserGroupsFound(), UserFindClose()

Esempio
```
Function void FindUsers()
    int P=UserFindFirst();
    bool Found;
    if UserNameFound(P)!="" then Found=true; else Found=false;end
    While(Found==true)
        MessageBox(UserNameFound(P),UserGroupsFound(P));
        Found=UserFindNext(P);
    end
    UserFindClose(P);
end
```

9.14 Recipes

9.14.1 RecipeCreate

Descrizione
Crea una ricetta in base al modello ed ai parametri stabiliti.

Sintassi
Bool RecipeCreate(String Model, String Recipe, Bool ErrorMsg)

Parametri
Model modello della ricetta (esistente)
Recipe nome della ricetta da creare
ErrorMsg mostra eventuale messaggio di errore

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
RecipeImport(), RecipeExecute()

Esempio
```
RecipeIsReady=RecipeCreate("Ricette di Produzione", "Produzione Filo Nero",
false);
```

9.14.2 RecipeExecute

Descrizione
Invia i valori relativi alla ricetta specificata ai vari dispositivi.

Sintassi

Bool RecipeExecute(String Model, String Recipe, Bool CompletionMsg)

Parametri

Model	modello di cui modificare i valori
Recipe	valori del modello
CompletionMsg	mostra il messaggio di completamento dell'esecuzione

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
RecipeCreate(), RecipeImport()

Esempio
```
RecipeExecute (CurrModel,NewInsertedValues, true)
```

9.14.3 RecipeImport

Descrizione
Salva i valori relativi al modello nella ricetta specificata.

Sintassi
Bool RecipeImport(String Model, String Recipe, Bool CompletionMsg)

Parametri

Model	modello di cui leggere i valori (esistente)
Recipe	valori del modello (esistente)
CompletionMsg	mostra il messaggio di completamento dell'importazione

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
RecipeCreate(), RecipeExecute()

Esempio
```
RecipeImport (CurrModel,CurrValues, false);
```

9.15 Report

9.15.1 ReportAppendRecord

Descrizione
Questo comando è usato per un report di tipo DAT configurato come "Salvataggio record su comando".Quando questa istruzione viene eseguita,un nuovo record viene scritto in coda al file del report; se il file non esiste viene automaticamente creato.
Se il report è configurato da salvare nella directory di default dei report,si può usare l'istruzione ReportCreate() per creare in automatico un nuovo file.In questo caso il file precedente verrà rinominato da (NomeReport).001 in (NomeReport).002 ed un nuovo file (NomeReport).001 verrà creato.

Sintassi
void ReportAppendRecord(String Name)

Parametri

Name nome del modello di report su cui operare

Valore restituito
-

Funzioni inerenti
ReportCreate()

Esempio
```
ReportAppendRecord("NomeReport");
```

9.15.2 ReportCreate

Descrizione
Elabora un file di Report.

Sintassi
Bool ReportCreate(String Name)

Parametri
Name nome del modello di report su cui basarsi per generare il report(esistente)

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
ReportDisplay(), ReportPrint(),ReportSetFullPathFileName()

Esempio
```
FileCreatedFlag= ReportCreate("CurrentSystemStatus");
```

9.15.3 ReportDisplay

Descrizione
Mostra il file dell'ultimo Report creato.
Questa funzione non è disponibile per report .DAT

Sintassi
Bool ReportDisplay(String Name)

Parametri
Name nome del file di report

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
ReportCreate(), ReportPrint()

Esempio
```
FileSuccesfullyReaded= ReportDisplay("SystemStatus");
```

9.15.4 ReportGetPeriodType

Descrizione:
Restituisce un intero che indica la frequenza con cui vengono prodotti i report

Sintassi:
Int ReportGetPeriodType(String ReportName)

Parametri:
ReportName nome del file di report

Valore restituito:
0 Mai (il report non viene generato)
1 Time (il report viene creato ogni intervallo di tempo specificato)
2 DayOfWeek (il report viene creato un giorno preciso della settimana)
3 DayOfMonth (il report viene creato un giorno preciso del mese)
4 DayAndMonth (il report viene creato in un giorno preciso nell'anno)

Funzioni inerenti:
ReportSetPeriodNone(), ReportSetPeriodTime(), ReportSetPeriodDayOfWeek(), ReportSetPeriodDayOfMonth(), ReportSetPeriodDayAndMonth()

Esempio
```
GroupAReportTime = ReportGetPeriodTime(GroupAReportFileName);
```

9.15.5 ReportInsertTemplate

Descrizione:
Inserisce l'immagine di un template in un report durante la sua elaborazione
(Funzione obsoleta sostituita da ReportInsertTemplateEx).
Questa funzione è da utilizzarsi esclusivamente all'interno di un report RTF/PDF.

Sintassi:
void ReportInsertTemplate (String text)

Parametri:
Text il nome del template la cui immagine verra inserita nel report

Valore restituito:
-

Funzioni inerenti:
ReportInsertText(),ReportInsertTemplateEx()

Esempio:
esempio di generazione di un report
N.B.: restrizioni sull'utilizzo
N.B.2: Le immagini dei template occupano molto spazio; l'immagine di un template con dimensioni pari a 1024x768 punti è di circa 2.5Mbytes per un report RTF.

9.15.6 ReportInsertTemplateEx

Descrizione:
Inserisce l'immagine di un template **completo** nel report durante la sua elaborazione
Questa funzione è da utilizzarsi esclusivamente all'interno di un report RTF/PDF.

Sintassi:

void ReportInsertTemplate (String text)

Parametri:
Text il nome del template la cui immagine verra inserita nel report

Valore restituito:
-

Funzioni inerenti:
ReportInsertText()

Esempio:
esempio di generazione di un report
N.B.: restrizioni sull'utilizzo
N.B.2: Le immagini dei template occupano molto spazio; l'immagine di un template con dimensioni pari a 1024x768 punti è di circa 2.5Mbytes per un report RTF..

9.15.7 ReportInsertHistoricalAlarmsRTF

Descrizione
Questa funzione inserisce automaticamente uno storico allarmi / eventi in un report in formato "RTF"

Sintassi

```
void ReportInsertHistoricalAlarmsRTF(
                                int sDay,
                                int sMonth,
                                int sYear,
                                int sHour,
                                int sMin,
                                int sSec,
                                int eDay,
                                int eMonth,
                                int eYear,
                                int eHour,
                                int eMin,
                                int eSec,
                                int ListType,
                                bool ShowDescription,
                                string ColumnName1,
                                string ColumnName2,
                                string ColumnName3,
                                string ColumnName4,
                                string ColumnName5,
                                string ColumnName6,
                                string ColumnName7,
                                string ColumnName8,
                                string ColumnName9,
                                string ColumnName10,
                                string ColumnName11,
                                string ColumnName12,
                                string ColumnName13,
                                int FilterClass1,
                                string FilterClass2,
                                string FilterClass3,
                                string FilterClass4,
                                string FilterClass5,
                                string FilterClass6,
```

```
                                    string FilterClass7)
```

Parametri	Descrizione
sDay = start day ***sMonth*** = start month ***sYear*** = start year ***sHour*** = start hour ***sMin*** = start minutes ***sSec*** = start seconds	Limite da cui iniziare ad inserire la lista allarmi / eventi nel report.
eDay = end day ***eMonth*** = end month ***eYear*** = end year ***eHour*** = end hour ***eMin*** = end minutes ***eSec*** = end seconds	Limite finale entro il quale inserire la lista allarmi / eventi nel report.
ListType	0: visualizza elenco allarmi 1: visualizza elenco eventi 2: visualizza sia allarmi che eventi
ShowDescription	True : visualizza l'intestazione delle colonne. False: non visualizza l'intestazione delle colonne
ColumName1 ***ColumName2*** ***ColumName3*** ***ColumName4*** ***ColumName5*** ***ColumName6*** ***ColumName7*** ***ColumName8*** ***ColumName9*** ***ColumName10*** ***ColumName11*** ***ColumName12*** ***ColumName13***	Ci possono essere al massimo 13 colonne.. Il tipo di dato da visualizzare in ogni colonna è specificato da*column name identifier* e può essere uno dei seguenti valori: "MESSAGE" : visualizza il messaggio di allarme/evento "START_DATE": visualizza data di inizio allarme/evento "START_TIME": visualizza ora di inizio allarme/evento "END_DATE": visualizza data di fine allarme/evento "END_TIME": visualizza ora di fine allarme/evento "DURATION": visualizza durata dell'allarme/evento "CLASS1": mostra classe 1 "CLASS2": mostra classe 2 "CLASS3": mostra classe 3 "CLASS4": mostra classe 4 "CLASS5": mostra classe 5 "CLASS6": mostra classe 6 "CLASS7": mostra classe 7 Le colonne specificate in ColumnName1..13 devono essere definite anche in ProjectManager->Configuration->Template->HistoricalAlarms o HistoricalEvents. Inserire una stringa nulla (**"")** se non si vuole visualizzare la colonna.
FilterClass1 ***FilterClass2*** ***FilterClass3*** ***FilterClass4*** ***FilterClass5*** ***FilterClass6*** ***FilterClass7***	Attraverso questi parametri è possibile specificare un filtro di visualizzazione per alarmi/eventi nel report. *FilterClass1* è un numero mentre *FilterClass2* ... *FilterClass7* sono stringhe di caratteri. *FilterClass1*= **-1** significa nessun filtro su *Class1*. *FilterClass2...FilterClass7*= **""** significa nessun

	filtro su Class2...Class7

Valore restituito
Nessuno.

Note:
Come impostare l'ampiezza delle colonne ?
Ogni colonna è separata da un carattere TAB, così nel file sorgente del report sarà sufficiente impostare la posizione del tabulatore di ogni colonna.
Per esempio con *Microsoft WordPad,* i tabulatori possono essere impostati selezionando il menu Format->Tabulator.

Esempio
```
ReportInsertHistoricalAlarmsRTF(
                                10,11,2005,0,0,0,
                                10,11,2005,23,59,59,
                                0,
                                true,
                                "MESSAGE",
                                "START_DATE",
                                "START_TIME",
                                "END_DATE",
                                "END_TIME",
                                "DURATION",
                                "CLASS1",
                                "CLASS2",
                                "CLASS5",
                                "",
                                "",
                                "",
                                "",
                                "",
                                -1,
                                "",
                                "",
                                "",
                                "",
                                "",
                                "",
                                "");
```

9.15.8 ReportInsertHistoricalAlarmsTXT

Descrizione
Questa funzione inserisce automaticamente uno storico allarmi / eventi in un report in formato "TXT"

Sintassi
```
void ReportInsertHistoricalAlarmsTXT(
                                int sDay,
                                int sMonth,
                                int sYear,
                                int sHour,
                                int sMin,
                                int sSec,
                                int eDay,
                                int eMonth,
                                int eYear,
                                int eHour,
```

```
int eMin,
int eSec,
int ListType,
bool ShowDescription,
string ColumnName1,
int ColumnWidth1,
string ColumnName2,
int ColumnWidth2,
string ColumnName3,
int ColumnWidth3,
string ColumnName4,
int ColumnWidth4,
string ColumnName5,
int ColumnWidth5,
string ColumnName6,
int ColumnWidth6,
string ColumnName7,
int ColumnWidth7,
string ColumnName8,
int ColumnWidth8,
string ColumnName9,
int ColumnWidth9,
string ColumnName10,
int ColumnWidth10,
string ColumnName11,
int ColumnWidth11,
string ColumnName12,
int ColumnWidth12,
string ColumnName13,
int ColumnWidth13,
int FilterClass1,
string FilterClass2,
string FilterClass3,
string FilterClass4,
string FilterClass5,
string FilterClass6,
string FilterClass7)
```

Parametri	Descrizione
sDay = start day ***sMonth*** = start month ***sYear*** = start year ***sHour*** = start hour ***sMin*** = start minutes ***sSec*** = start seconds	Limite da cui iniziare ad inserire la lista allarmi / eventi nel report.
eDay = end day ***eMonth*** = end month ***eYear*** = end year ***eHour*** = end hour ***eMin*** = end minutes ***eSec*** = end seconds	Limite finale entro il quale inserire la lista allarmi / eventi nel report.
ListType	0: visualizza elenco allarmi 1: visualizza elenco eventi 2: visualizza sia allarmi che eventi
ShowDescription	True : visualizza l'intestazione delle colonne. False: non visualizza l'intestazione delle colonne

ColumName1 ***ColumnWidth1*** ***ColumName2*** ***ColumnWidth2*** ***ColumName3*** ***ColumnWidth3*** ***ColumName4*** ***ColumnWidth4*** ***ColumName5*** ***ColumnWidth5*** ***ColumName6*** ***ColumnWidth6*** ***ColumName7*** ***ColumnWidth7*** ***ColumName8*** ***ColumnWidth8*** ***ColumName9*** ***ColumnWidth9*** ***ColumName10*** ***ColumnWidth10*** ***ColumName11*** ***ColumnWidth11*** ***ColumName12*** ***ColumnWidth12*** ***ColumName13***. ***ColumnWidth13***	Ci possono essere al massimo 13 colonne.. Il tipo di dato da visualizzare in ogni colonna è specificato da *column name identifier* e può essere uno dei seguenti valori: "MESSAGE" : visualizza il messaggio di allarme/evento "START_DATE": visualizza data di inizio allarme/evento "START_TIME": visualizza ora di inizio allarme/evento "END_DATE": visualizza data di fine allarme/evento "END_TIME": visualizza ora di fine allarme/evento "DURATION": visualizza durata dell'allarme/evento "CLASS1": mostra classe 1 "CLASS2": mostra classe 2 "CLASS3": mostra classe 3 "CLASS4": mostra classe 4 "CLASS5": mostra classe 5 "CLASS6": mostra classe 6 "CLASS7": mostra classe 7 Le colonne specificate in ColumnName1..13 devono essere definite anche in ProjectManager->Configuration->Template->HistoricalAlams o HistoricalEvents. Inserire una stringa nulla (**""**) se non si vuole visualizzare la colonna. *Column Width identifier* specifica il numero massimo di caratteri per ogni colonna.
FilterClass1 ***FilterClass2*** ***FilterClass3*** ***FilterClass4*** ***FilterClass5*** ***FilterClass6*** ***FilterClass7***	Attraverso questi parametri è possibile specificare un filtro di visualizzazione per alarmi/eventi nel report. *FilterClass1* è un numero mentre *FilterClass2* ... *FilterClass7* sono stringhe di caratteri. *FilterClass1*= **-1** significa nessun filtro su *Class1*. *FilterClass2...FilterClass7*= **""** significa nessun filtro su Class2...Class7.

Valore restituito
Nessuno.

Esempio

```
ReportInsertHistoricalAlarmsTXT(
                                10,11,2005,0,0,0,
                                10,11,2005,23,59,59,
                                0,
                                true,
                                "MESSAGE",30,
                                "START_DATE",13,
                                "START_TIME",13,
                                "END_DATE",13,
                                "END_TIME",13,
                                "DURATION",15,
```

```
                                        "CLASS1",3,
                                        "CLASS2",10,
                                        "CLASS5",10,
                                        "",0,
                                        "",0,
                                        "",0,
                                        "",0,
                                        -1,
                                        "",
                                        "",
                                        "",
                                        "",
                                        "",
                                        "",
                                        "");
```

9.15.9 ReportInsertText

Descrizione:
Inserisce del testo nel report durante la sua elaborazione.
Questa funzione è da utilizzarsi esclusivamente all'interno di un report TXT e RTF/PDF.

Sintassi:
void ReportInsertText (String text)

Parametri:
Text la stringa che verra inserita nel report

Valore restituito:
-

Funzioni inerenti:
ReportInsertTemplate()

Esempio:
esempio di generazione di un report
N.B.: restrizioni sull'utilizzo

9.15.10 ReportInsertUserChangesRTF

Descrizione
Questa funzione inserisce automaticamente uno storico interventi operatore in un report in formato "RTF"

Sintassi

```
void ReportInsertUserChangesRTF(
                                    int sDay,
                                    int sMonth,
                                    int sYear,
                                    int sHour,
                                    int sMin,
                                    int sSec,
                                    int eDay,
                                    int eMonth,
                                    int eYear,
                                    int eHour,
                                    int eMin,
                                    int eSec,
```

```
                                bool ShowDescription,
                                string ColumnName1,
                                string ColumnName2,
                                string ColumnName3,
                                string ColumnName4,
                                string ColumnName5)
```

Parametri	Descrizione
sDay = start day ***sMonth*** = start month ***sYear*** = start year ***sHour*** = start hour ***sMin*** = start minutes ***sSec*** = start seconds	Limite da cui iniziare ad inserire la lista interventi operatore nel report.
eDay = end day ***eMonth*** = end month ***eYear*** = end year ***eHour*** = end hour ***eMin*** = end minutes ***eSec*** = end seconds	Limite finale entro il quale inserire la lista interventi operatore nel report.
ShowDescription	True : visualizza l'intestazione delle colonne. False: non visualizza l'intestazione delle colonne
ColumnName1 ***ColumnName2*** ***ColumnName3*** ***ColumnName4*** ***ColumnName5***	Ci possono essere al massimo 5 colonne.. Il tipo di dato da visualizzare in ogni colonna è specificato da*column name identifier* e può essere uno dei seguenti valori: "CODE" : visualizza identificatore operazione "USER": visualizza nome utente che ha eseguito l'operazione "DATE": visualizza data in cui è stata effettuata l'operazione "TIME": visualizza ora in cui è stata effettuata l'operazione "MESSAGE": visualizza descrizione dell'operazione Inserire una stringa nulla (**""**) se non si vuole visualizzare la colonna.

Valore restituito
Nessuno.

Note:
Come impostare l'ampiezza delle colonne ?
Ogni colonna è separata da un carattere TAB, così nel file sorgente del report sarà sufficiente impostare la posizione del tabulatore di ogni colonna.
Per esempio con *Microsoft WordPad,* i tabulatori possono essere impostati selezionando il menu Format->Tabulator.

Esempio

```
ReportInsertUserChangesRTF(
                                10,11,2005,0,0,0,
                                10,11,2005,23,59,59,
```

```
                                    true,
                                    "CODE",
                                    "USER",
                                    "DATE",
                                    "TIME",
                                    "MESAGE");
```

9.15.11 ReportInsertUserChangesTXT

Descrizione
Questa funzione inserisce automaticamente uno storico interventi operatore in un report in formato "TXT"

Sintassi

```
void ReportInsertUserChangesTXT(
                                    int sDay,
                                    int sMonth,
                                    int sYear,
                                    int sHour,
                                    int sMin,
                                    int sSec,
                                    int eDay,
                                    int eMonth,
                                    int eYear,
                                    int eHour,
                                    int eMin,
                                    int eSec,
                                    bool ShowDescription,
                                    string ColumnName1,
                                    int ColumnWidth1,
                                    string ColumnName2,
                                    int ColumnWidth2,
                                    string ColumnName3,
                                    int ColumnWidth3,
                                    string ColumnName4,
                                    int ColumnWidth4,
                                    string ColumnName5,
                                    int ColumnWidth5)
```

Parametri	Descrizione
sDay = start day ***sMonth*** = start month ***sYear*** = start year ***sHour*** = start hour ***sMin*** = start minutes ***sSec*** = start seconds	Limite da cui iniziare ad inserire la lista interventi operatore nel report.
eDay = end day ***eMonth*** = end month ***eYear*** = end year ***eHour*** = end hour ***eMin*** = end minutes ***eSec*** = end seconds	Limite finale entro il quale inserire la lista interventi operatore nel report.
ShowDescription	True : visualizza l'intestazione delle colonne. False: non visualizza l'intestazione delle colonne

ColumName1 ***ColumnWidth1*** ***ColumName2*** ***ColumnWidth2*** ***ColumName3*** ***ColumnWidth3*** ***ColumName4*** ***ColumnWidth4*** ***ColumName5*** ***ColumnWidth5***	Ci possono essere al massimo 5 colonne.. Il tipo di dato da visualizzare in ogni colonna è specificato da *column name identifier* e può essere uno dei seguenti valori: "CODE" : visualizza identificatore operazione "USER": visualizza nome utente che ha eseguito l'operazione "DATE": visualizza data in cui è stata effettuata l'operazione "TIME": visualizza ora in cui è stata effettuata l'operazione "MESSAGE": visualizza descrizione dell'operazione Inserire una stringa nulla (**""**) se non si vuole visualizzare la colonna. *Column Width identifier* specifica il numero massimo di caratteri per ogni colonna.

Valore restituito
Nessuno.

Esempio
```
ReportInsertUserChangesTXT(
                                10,11,2005,0,0,0,
                                10,11,2005,23,59,59,
                                true,
                                "CODE",15,
                                "USER",12,
                                "DATE",10,
                                "TIME",10,
                                "MESSAGE",40);
```

9.15.12 ReportLotTime

Descrizione:
Restituisce una stringa formattata nel seguente modo:
GG/MM/AAAA - HH:MN:SS
la stringa è costruita in base ai parametri

Sintassi:
String ReportLotTime (Int GG, Int MM, Int AAAA, Int HH, Int MN, Int SS)

Parametri:
GG giorno
MM mese
AAAA anno
HH ora
MN minuti
SS secondi

Valore restituito:
i parametri formattati secondo la stringa

Funzioni inerenti:
-

Esempio
```
String start;
start=ReportLotTime (GetDayOfMonth(), GetMonth(), GetYear(),  0, 0, 0);
```

9.15.13 ReportPrint

Descrizione
Stampa il file dell'ultimo Report creato.
Questa funzione non è disponibile per report .DAT

Sintassi
Bool ReportPrint(String Name, Bool ViewPrintDlg)

Parametri
Name nome del file di report
ViewPrintDlg true (mostra le preferenze della stampante prima di stampare) false (non le mostra)

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
ReportCreate(), ReportDisplay()

Esempio
```
FileSuccesfullyPrinted= ReportPrint("SystemStatus",true);
```

9.15.14 ReportShowCreationList

Descrizione
Visualizza la lista dei report da elaborare

Sintassi
Void ReportShowCreationList()

Parametri
-

Valore restituito
-

Funzioni inerenti
ReportShowHistList()

Esempio
```
ReportShowCreationList();
```

9.15.15 ReportShowHistList

Descrizione
Visualizza la lista dei report storici

Sintassi

Void ReportShowHistList()

Parametri
-

Valore restituito
-

Funzioni inerenti
ReportShowHistList()

Esempio
```
ReportShowHistList();
```

9.15.16 ReportSetFullPathFileName

Descrizione
Specifica il nome (completo di percorso assoluto) che dovrà essere associato al prossimo report che verrà creato.
Questa istruzione è di solito seguita da ReportCreate()

Sintassi
Bool ReportSetFullPathFileName(String ReportType,String FullPathFileName)

Parametri

ReportType	nome del modello di report (esistente)
FullPathFileName	nome da associare al prossimo report di tipo ReportType che verrà creato. Il nome del file deve essere completo di percorso assoluto,ad esempio: "C:\DOCUMENTS\Report_07_10_1970.txt". Il percorso deve comunque esistere altrimenti la funzione ReportCreate() non creerà alcun report.Se il report con il nome specificato è già presente,esso verrà sovrascritto.Se è stato associato il nome del file al tipo di report,tramite ReportSetFullPathFileName(),tutti i report di quel tipo che verranno creati sovrascriveranno il file specificato; per tornare al funzionamento standard (creazione file nella directory del tipo di report) è necessario usare di nuovo l'istruzione ReportSetFullPathFileName() specificando "" (Nulla) nel parametro FullPathFileName.

Valore restituito
true (operazione si è conclusa con successo)
false (in caso contrario)

Funzioni inerenti
ReportCreate()

Esempio
```
//=====================================================
 // Associa al modello "ProductionLot" il nome "C:\Documents\LotNumber1.rtf"
//=====================================================
 if
(ReportSetFullPathFileName("ProductionLot","C:\Documents\LotNumber1.rtf")==
true) then

      //================================
      // Crea il report C:\Documents\LotNumber1.rtf
      //================================
      ReportCreate("ProductionLot");
```

```
        //==============================
        // Ritorna al funzionamento standard del report
        //==============================
        ReportSetFullPathFileName("ProductionLot","");
end
```

9.15.17 ReportSetPeriodDayOfWeek

Descrizione:
Specifica quale giorno della settimana creare il report

Sintassi:
Bool ReportSetPeriodDayOfWeek (String ReportName,Int Giorno,Int OraSincron, Int MinSincron, Int SecSincron)

Parametri:

ReportName	nome del file di report
Giorno	giorno della settimana (1=lun, 2=mar, 3=merc...)
OraSincron, MinSincron, SecSincron	specificano l'ora da cui partire

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso il file di report non esista oppure l'intervallo di tempo specificato non sia corretto)

Funzioni inerenti:
ReportGetPeriodType(), ReportSetPeriodNone(), ReportSetPeriodTime(), ReportSetPeriodDayOfMonth(), ReportSetPeriodDayAndMonth()

Esempio:
`ReportSetPeriodDayOfWeek(RepName,2,9,0,0);` // mar. dalle 9.00

9.15.18 ReportSetPeriodDayOfMonth

Descrizione:
Specifica quale giorno del mese creare il report

Sintassi:
Bool ReportSetPeriodDayOfMonth (String ReportName,Int Giorno,Int OraSincron, Int MinSincron, Int SecSincron)

Parametri:

ReportName	nome del file di report
Giorno	giorno del mese
OraSincron, MinSincron, SecSincron	specificano l'ora da cui partire

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso il file di report non esista oppure l'intervallo di tempo specificato non sia corretto)

Funzioni inerenti:
ReportGetPeriodType(), ReportSetPeriodNone(), ReportSetPeriodTime(), ReportSetPeriodDayOfWeek(), ReportSetPeriodDayAndMonth()

Esempio:
`ReportSetPeriodDayOfMonth(RepName,24,9,0,0);` // giorno 24 dalle 9

9.15.19 ReportSetPeriodDayAndMonth

Descrizione:
Specifica un momento nell'anno in cui creare il report

Sintassi:
Bool ReportSetPeriodDayAndMonth (String ReportName,Int Giorno, Int Mese,Int OraSincron, Int MinSincron, Int SecSincron)

Parametri:

ReportName	nome del file di report
Giorno	giorno del mese
Mese	mese
OraSincron, MinSincron, SecSincron	specificano l'ora da cui partire

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso il file di report non esista oppure l'intervallo di tempo specificato non sia corretto)

Funzioni inerenti:
ReportGetPeriodType(), ReportSetPeriodNone(), ReportSetPeriodTime(), ReportSetPeriodDayOfWeek(), ReportSetPeriodDayOfMonth()

Esempio:
```
ReportSetPeriodDayOfMonth(RepName,24,1,9,0,0); // 24 Genn. dalle 9
```

9.15.20 ReportSetPeriodNone

Descrizione:
Si specifica che il report non deve essere creato

Sintassi:
Bool ReportSetPeriodNone (String ReportName)

Parametri:

ReportName	nome del file di report

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso il file di report non esista oppure l'intervallo di tempo specificato non sia corretto)

Funzioni inerenti:
ReportGetPeriodType(), ReportSetPeriodTime(), ReportSetPeriodDayOfWeek(), ReportSetPeriodDayOfMonth(), ReportSetPeriodDayAndMonth()

Esempio:
```
ReportSetPeriodNone(GroupAReportFileName);
```

9.15.21 ReportSetPeriodTime

Descrizione:
Specifica l'intervallo di tempo per la creazione del report

Sintassi:
Bool ReportSetPeriodTime (String ReportName,Int OraPeriodo, Int MinPeriodo, Int SecPeriodo,Int OraSincron, Int MinSincron, Int SecSincron)

Parametri:

ReportName	nome del file di report
OraPeriodo, MinPeriodo, SecPeriodo	specificano l'intervallo di tempo tra i report
OraSincron, MinSincron, SecSincron	specificano l'ora da cui partire

Valore restituito:
true (in caso l'operazione venga effettuata con successo)
false (in caso il file di report non esista oppure l'intervallo di tempo specificato non sia corretto)

Funzioni inerenti:
ReportGetPeriodType(), ReportSetPeriodNone(), ReportSetPeriodDayOfWeek(), ReportSetPeriodDayOfMonth(), ReportSetPeriodDayAndMonth()

Esempio:
`ReportSetPeriodTime(RepName,1,0,0,9,0,0);` // ogni ora dalle 9.00

9.15.22 ReportInsertText ReportInsertTemplate example

Supponiamo di dover generare un rapporto giornaliero contenente le anomalie riscontrate durante la giornata.
Il file contenente le anomalie è un file di testo che contiene in ogni riga il momento (data e ora) in cui è avvenuta.

ANOMALIE.TXT:

....
1/2/2001 09:16:58
1/2/2001 09:19:26
1/2/2001 09:21:39
1/2/2001 09:21:59
1/2/2001 09:22:19
1/2/2001 09:22:40
1/2/2001 09:23:00
....

Questo è il file che descrive come è fatto il report
ANOMALIE.RTF:

ANOMALIE RISCONTRATE:

{InserisciAnomalie()}

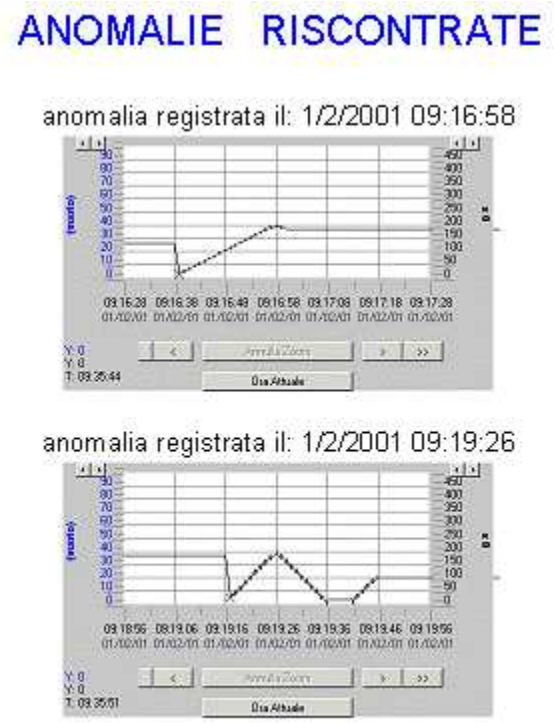
Questo è il codice della funzione richiamata dal report:
REPORT.WLL:

```
Function void InserisciAnomalie()
 string line;
 int hh; int mn;  int ss; int yy; int mm; int dd;
 int h1; int m1; int s1; int h2; int m2; int s2;
 int a;
 int i=0;
 int h=FileOpen("e:\anomalie.txt","rt");
 While (FileEOF(h) == 0)
  line=FileReadLn(h);
```

```
 i=i+1;
 if (FileEOF(h) == 0) then
     ReportInsertText("anomalia registrata il: "+line+Eol());
     // decodifica la data e ora dell'anomalia
     a=StrPos(line,"/"); dd=StrToInt(StrSubString(line,1,a-1)); line=StrDelete(line,1,a);
     a=StrPos(line,"/"); mm=StrToInt(StrSubString(line,1,a-1)); line=StrDelete(line,1,a);
     a=StrPos(line," "); yy=StrToInt(StrSubString(line,1,a-1)); line=StrDelete(line,1,a);
     a=StrPos(line,":"); hh=StrToInt(StrSubString(line,1,a-1)); line=StrDelete(line,1,a);
     a=StrPos(line,":"); mn=StrToInt(StrSubString(line,1,a-1)); line=StrDelete(line,1,a);
     // dobbiamo creare un intervallo di tempo che parte 30" prima e termina 30"dopo
     ss=StrToInt(line);
     s1=ss-30; m1=mn; h1=hh;
     if (s1<0) then
        s1=60+s1;  m1=m1-1;
        if (m1==-1) then
           m1=59; h1=h1-1;
           if (h1<0) then
              h1=23;
           end
        end
     end
     s2=s1;
     m2=m1+1;
     h2=h1;
     if (m2==60) then
           m2=0; h2=h2+1;
           if (h2==24) then
              h2=0;
           end
     end

     ChartSetTimeRange(dd,mm,yy,h1,m1,s1,dd,mm,yy,h2,m2,s2); //da -30" a +30"
     // inserisci il grafico dell anomalia
     ReportInsertTemplateEx("Grafico");
     // il template inserito nella riga precedente 'Grafico'
     // contiente unicamente un oggetto chart che rappresenta la porta su cui si riscontrano le anomalie
     ReportInsertText(Eol()+Eol()+Eol()+Eol());
  end
 end
 FileClose(h);
end
```

Il report risultante alla fine della giornata
REPORT.001:



9.15.23 Report: Note

Per utilizzare le nuove funzioni di manipolazione del report:

```
ReportInsertText()
ReportInsertTemplate()
```

si deve possedere una libreria (riched32.dll) presente nel sistema quando è installato almeno uno dei seguenti software:

MS Windows NT 4.0 o versioni successive
MS Internet Explorer 4.0 o versioni successive
MS Office 97 o versioni successive

9.16 SMS

9.16.1 SMSCloseChannel

Descrizione
Chiude la comunicazione verso il modem GSM specificato

Sintassi
`Bool SMSCloseChannel(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
True operazione eseguita con esito positivo
False errore - probabilemnte la comunicazione con il modem specificato non era ancora

stata aperta.

Funzioni inerenti
SMSOpenChannel()

Esempio
```
Function void TestSignalQuality()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        MessageBox(SMSGetSignalQuality(ComHandle),"SMSSignalQuality");
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end
```

9.16.2 SMSDelete

Descrizione
Cancella dalla SIM card l SMS specificato

Sintassi
`Int SMSDelete(Int Handle, Int RecordIndex)`

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.
int RecordIndex	RecordIndex associato all' SMS da cancellare.

Valore restituito

0	il modem ha risposto "OK"
-1	impossibile aprire la porta seriale
-3	il modem ha risposto "ERROR"
-4	il modem ha risposto in maniera non standard (cioè diverso da "OK" o "ERROR")
-5	il modem non risponde

Funzioni inerenti
SMSOpenChannel(), SMSCloseChannel(), SMSFindFirst()

Esempio
```
Function void TestSMSDelete()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        SMSDelete(comHandle,1);
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end
```

9.16.3 SMSFindClose

Descrizione
Termina la scansione dell'elenco degli SMS ricevuti, creato con l'istruzione SMSFindFirst() e libera la memoria allocata.

Sintassi
`Bool SMSFindClose(Int Handle)`

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.

Valore restituito

True	Operazione completata con successo.
False	E' stato specificato un handle non valido.

Funzioni inerenti

SMSOpenChannel(), SMSCloseChannel(), SMSFindFirst(), SMSFindNext(), SMSFoundRecordIndex() , SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(),SMSFoundIsValid(), SMSDelete()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.4 SMSFindFirst

Descrizione

Legge dalla SIM card l'elenco degli SMS ricevuti, li ordina in modo "Ascendente" o "Discendente", e punta il primo record logico.

Sintassi

```
Int SMSFindFirst(Int Handle, Int Mode, Bool Sort)
```

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.
Int Mode	può essere uno dei seguenti casi: 0 : messaggi ricevuti e non ancora letti 1 : messaggi ricevuti e già letti 2 : messaggi caricati ma non inviati 3 : messaggi caricati e inviati

4 : tutti i messaggi

Bool Sort — true: l'elenco degli SMS verrà ordinato in modo "Ascendente" (dal meno recente al più recente)
false:l'elenco degli SMS verrà ordinato in modo "Discendente" (dal più recente al meno recente)

Valore restituito

> 0 — Numero di record SMS trovati.
I campi dell' SMS puntato possono essere letti tramite SMSFoundRecordIndex(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage()
Usare SMSFindNext() per scandire l'elenco degli SMS .
Usere SMSFindClose() per terminare la scansione.

0 — il modem ha risposto "OK"ma non è stato trovato alcun SMS.
Non è necessario usare SMSFindClose() per terminare la scansione.

-1 — impossibile aprire la porta seriale

-3 — il modem ha risposto "ERROR"

-4 — il modem ha risposto in maniera non standard (cioè diverso da "OK" o "ERROR")

-5 — il modem non risponde

Funzioni inerenti

SMSOpenChannel(), SMSCloseChannel(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(), SMSFoundIsValid(), SMSDelete()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.5 SMSFindNext

Descrizione

Sposta il puntatore sul record logico successivo nella lista degli SMS ricevuti, precedentemente letta con l'istruzione SMSFindFirst().

Sintassi
`Bool SMSFindNext(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
True E' stato trovato il record SMS successivo.
I campi dell' SMS puntato possono essere letti tramite SMSFoundRecordIndex(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundCallerNumber(), SMSFoundCallerID(), SMSFoundMessage()
Usare SMSFindNext() per scandire l'elenco degli SMS .
Usere SMSFindClose() per terminare la scansione.
False E' stata raggiunta la fine della lista degli SMS ricevuti o è stato specificato un handle non valido.

Funzioni inerenti
SMSOpenChannel(), SMSCloseChannel(), SMSFindFirst(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(), SMSFoundIsValid(),SMSDelete()

Esempio
```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.6 SMSFoundIsValid

Descrizione
Ritorna true se l'elemento attualmente puntato, nell'elenco degli SMS ricevuti, rappresenta un SMS supportato (cioè Testo semplice, con alfabeto GSM a 7 bit e codifica SMS).
Non sono supportati gli SMS concatenati.

Sintassi
`Bool SMSFoundIsValid(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
True SMS supportato
False SMS non supportato.

Funzioni inerenti
SMSOpenChannel(), SMSCloseChannel(), SMSFindFirst(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(), SMSDelete()

Esempio
```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.7 SMSFoundMessage

Descrizione
Ritorna il messaggio dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`String SMSFoundMessage(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
Messaggio SMS

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp()

, SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(),SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.8 SMSFoundRecordIndex

Descrizione

Ritorna il "RecordIndex" fisico dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi

```
Int SMSFoundRecordIndex(Int Handle)
```

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.

Valore restituito

SMS Record Index

Funzioni inerenti

SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundTimeStamp(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(),SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
```

```
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.9 SMSFoundRecordType

Descrizione
Ritorna il "RecordType" dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`String SMSFoundRecordType(Int Handle)`

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.

Valore restituito

"REC UNREAD"	messaggio ricevuto e non ancora letto
"REC READ"	messaggio ricevuto e già letto

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp() , SMSFoundTimeStampString(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(),SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.10 SMSFoundSenderID

Descrizione
Ritorna l'identificatore del mittente dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`String SMSFoundSenderID(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
Identificatore del mittente.

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp() , SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundMessage(),SMSFoundIsValid()

Esempio
```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.11 SMSFoundSenderNumber

Descrizione
Ritorna il numero telefonico del mittente dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`String SMSFoundSenderNumber(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
Numero mittente SMS

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp() , SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderID(), SMSFoundMessage(),SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.12 SMSFoundTimeStamp

Descrizione
Ritorna il "TimeStamp" dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`Unsigned SMSFoundTimeStamp(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
SMS time stamp espresso in numero di secondi trascorsi dal 1 gennaio 1901.

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStampString(), SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage(),SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.13 SMSFoundTimeStampString

Descrizione
Ritorna il "TimeStamp" in formato stringa dell'elemento attualmente puntato, nell'elenco degli SMS ricevuti.

Sintassi
`String SMSFoundTimeStampString(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito
SMS time stamp senza alcuna conversione.

Funzioni inerenti
SMSFindFirst(), SMSFindNext(), SMSFindClose(), SMSFoundRecordIndex(), SMSFoundTimeStamp() , SMSFoundRecordType(), SMSFoundSenderNumber(), SMSFoundSenderID(), SMSFoundMessage() ,SMSFoundIsValid()

Esempio

```
Function void GetSMSList()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        if (SMSFindFirst(ComHandle,4,true)>0) then
            ViewSMSData(ComHandle);
            while (SMSFindNext(ComHandle)==true) then
                ViewSMSData(ComHandle);
            end
            SMSFindClose(ComHandle);
        end
        SMSCloseChannel(ComHandle);
    else
```

```
        MessageBox(ComHandle,"Error code");
    end
end

Function void ViewSMSData(int ComHandle)
   MessageBox(SMSFoundRecordIndex(ComHandle),"RecodIndex");
   MessageBox(SMSFoundIsValid(ComHandle),"Is valid SMS ?");
   MessageBox(SMSFoundTimeStampString(ComHandle),"SMS TimeStampString");
   MessageBox(SMSFoundTimeStamp(ComHandle),"SMS TimeStamp");
   MessageBox(SMSFoundRecordType(ComHandle),"Record Type");
   MessageBox(SMSFoundSenderNumber(ComHandle),"Caller number");
   MessageBox(SMSFoundSenderID(ComHandle),"Caller ID");
   MessageBox(SMSFoundMessage(ComHandle),"SMS Message");
end
```

9.16.14 SMSGetSignalQuality

Descrizione
Ritorna l'indicazione della potenza del segnale ricevuto dal modem GSM.

Sintassi
`Int SMSGetSignalQuality(Int Handle)`

Parametri
Int Handle handle associato alla porta seriale connessa al modem GSM.

Valore restituito

0	-113 dBm o meno
1	-111 dBm
2..30	-109...-53 dBm
31	-51 dBm o maggiore
99	sconosciuto o non determinabile
-1	impossibile aprire la porta seriale
-3	il modem ha risposto "ERROR"
-4	il modem ha risposto in maniera non standard (cioè diverso da "OK" o "ERROR")
-5	il modem non risponde

Funzioni inerenti
SMSOpenChannel(), SMSCloseChannel()

Esempio

```
Function void TestSignalQuality()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        MessageBox(SMSGetSignalQuality(ComHandle),"SMSSignalQuality");
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end
```

9.16.15 SMSOpenChannel

Descrizione
Opre la comunicazione con un modem GSM

Sintassi
`Int SMSOpenChannel(Int COMx, Int Speed, String PIN, String ServiceCentreAddress)`

Parametri
Int COMx numero della porta seriale associata al modem GSM .

Int Speed	velocità di comunicazione della porta seriale (1200, 2400, 4800, 9600, 19200, 38400, 57600,115200).
String PIN	numero PIN usato per accedere alla SIM. Specificare "" per non effettuare l'impostazione del PIN.
String ServiceCentreAddress	numero del centro servizi. E' possibile reperire il numero del crento servizi inserendo la SIM in un telefono mobile e cercando fra i menu di impostazione.Tale numero può essere richiesto direttamente al fornitore del servizion o può essere trovato sulla guida della SIM card.

Valore restituito

Handle	se è un valore >0 allora rappresenta il numerod i handle associato alla porta COM appena aperta.
-1	impossibile aprire la porta seriale
-3	il modem ha risposto "ERROR"
-4	il modem ha risposto in maniera non standard (cioè diverso da "OK" o "ERROR")
-5	il modem non risponde
-11	il numero del centro servizi supera i 40 caratteri

Funzioni inerenti
SMSCloseChannel(), SMSSend(), SMSDelete(), SMSGetSignalQuality(), SMSFindFirst()

Esempio
```
Function void TestSignalQuality()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        MessageBox(SMSGetSignalQuality(ComHandle),"SMSSignalQuality");
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end
```

9.16.16 SMSSend

Descrizione
Invia un SMS al numero destinazione specificato
E' usata la codifica standard per messaggi GSM conl'alfabeto di default a 7 bit GMS 03.38.
Non sono supportati gli SMS concatenati.
La lunghezza massima consentita per il testo del messagggio è 160 caratteri.

Sintassi
```
Int SMSSend(Int Handle, string Number, string Message)
```

Parametri

Int Handle	handle associato alla porta seriale connessa al modem GSM.
String Number	numero telefonico dell'apparecchio destinatario
String Message	testo del messagio da inviare

Valore restituito

0	il modem ha risposto "OK"
-1	impossibile aprire la porta seriale
-3	il modem ha risposto "ERROR"

-4 il modem ha risposto in maniera non standard (cioè diverso da "OK" o "ERROR")
-5 il modem non risponde
-11 il numero destinatario supera i 40 caratteri
-12 il messaggio supera i 160 caratteri

Funzioni inerenti
SMSOpenChannel(), SMSCloseChannel(), SMSDelete(), SMSGetSignalQuality(), SMSFindFirst()

Esempio
```
Function void TestSMSSend()
    int ComHandle = SMSOpenChannel(1,9600,"","");
    if (ComHandle>0) then
        SMSSend(   comHandle,
                   GetStrGateValue("DestNumber",0),
                   GetStrGateValue("Message",0));
        SMSCloseChannel(ComHandle);
    else
        MessageBox(ComHandle,"Error code");
    end
end
```

9.17 String

9.17.1 CharToStr

Descrizione
Crea una stringa formata da un solo carattere di codice ascii specificato.

Sintassi
String CharToStr(Int AsciiCode)

Parametri
AsciiCode codice ascii del carattere con cui verra creata la stringa

Valore restituito
la stringa contenente il carattere richiesto

Funzioni inerenti
Eol(), StrToReal(), StrToInt(), IntToStr(), RealToStr(), StrToChar()

Esempio
```
FGender=CharToStr(70);
```

9.17.2 Eol

Descrizione
Restituisce una stringa contenente il carattere di fine linea (End Of Line).

Sintassi
String Eol()

Parametri
-

Valore restituito
la stringa contenente il carattere di fine linea

Funzioni inerenti
CharToStr()

Esempio
```
Linea=Eol();
```

9.17.3 IntToStr

Descrizione
Converte un numero intero in stringa

Sintassi
String IntToStr(Int Value)

Parametri
Value valore intero da convertire

Valore restituito
il valore in formato stringa

Funzioni inerenti
StrToInt(), StrToReal(), RealToStr(), ChatToStr()

Esempio
```
Anno=IntToStr(1975);
```

9.17.4 RealToStr

Descrizione
Converte un numero reale in stringa

Sintassi
String RealToStr(Real Value)

Parametri
Value valore reale da convertire

Valore restituito
il valore in formato stringa

Funzioni inerenti
StrToInt(), StrToReal(), IntToStr(), ChatToStr()

Esempio
```
AbsZero=RealToStr(-273.5);
```

9.17.5 StrCase

Descrizione
Converte una stringa in formato maiuscolo o minuscolo

Sintassi
String StrCase(String S, Bool UpperCased)

Parametri

S	la stringa da elaborare
UpperCased	true (Maiuscolo)

false (minuscolo)

Valore restituito
la stringa maiuscola/minuscola

Funzioni inerenti
-

Esempio
```
LowCase=StrCase("Valore",false);
```
// LowCase= "valore"

9.17.6 StrConcat

Descrizione
Concatena due stringhe

Sintassi
String StrConcat(String S1, String S2)

Parametri

S1	prima stringa
S2	seconda stringa

Valore restituito
una stringa contenente l'unione di entrambi

Funzioni inerenti
StrDelete()

Esempio
```
Dato=StrConcat ("Nome ","Marco");
```
// Dato = "Nome Marco"

9.17.7 StrDelete

Descrizione
Cancella una sequenza di caratteri in una stringa

Sintassi
String StrDelete(String S, Int Start, Int Len)

Parametri

S	stringa da cui rimuovere i caratteri
Start	posizione nella stringa in cui parte la sequenza da rimuovere
Len	lunghezza della sequenza

Valore restituito
la stringa passata come parametro da cui è stata eliminata la sottostringa specificata

Funzioni inerenti
StrSubString(), StrPos()

Esempio
```
Res = StrDelete("Oggi non c'è il sole",6,3);
```
// Res = "Oggi c'è il sole"

9.17.8 StrLen

Descrizione
Restituisce la lunghezza di una stringa.

Sintassi
Int StrLen(String Value)

Parametri
Value la stringa da analizzare

Valore restituito
la lunghezza della stringa

Funzioni inerenti
-

Esempio
```
LungNome = StrLen(Nome);
```

9.17.9 StrOfChar

Descrizione
Crea una stringa costituita dallo stesso carattere ripetuto più volte

Sintassi
String StrOfChar(Int Chr, int Len)

Parametri
Chr il carattere da ripetere
Len lunghezza della stringa

Valore restituito
la stringa creata con le specifiche

Funzioni inerenti
-

Esempio
```
TenB = StrOfChar (66,10);            // = "BBBBBBBBBB"
```

9.17.10 StrPos

Descrizione
Ricerca una stringa in un'altra

Sintassi
Int StrPos(String S, String SubStr)

Parametri
S stringa in cui effettuare la ricerca
SubStr sottostringa da ricercare

Valore restituito
Posizione della sottostringa in caso questa venga rilevata, altrimenti 0

Funzioni inerenti

StrDelete(), StrSubString()

Esempio
```
At = StrPos("Oggi non c'è il sole","non"); // At = 6
```

9.17.11 StrSubString

Descrizione
Estrae una sequenza di caratteri da una stringa

Sintassi
String StrSubString(String S, Int Start, Int Len)

Parametri

S	stringa da cui estrarre i caratteri
Start	posizione nella stringa in cui parte la sequenza
Len	lunghezza della sequenza

Valore restituito
sottostringa specificata nei parametri

Funzioni inerenti
StrDelete(),StrPos()

Esempio
```
Res = StrSubString("Oggi non c'è il sole",6,3); // Res = "non"
```

9.17.12 StrToChar

Descrizione
Restituisce il carattere alla posizione specificata in una stringa.

Sintassi
Int StrToChar (String Str, Int Pos)

Parametri

Str	stringa da cui estrarre il carattere
Pos	posizione del carattere nella stringa (1 rappresenta il primo carattere)

Valore restituito
il carattere richiesto della stringa

Funzioni inerenti
Eol(), StrToReal(), StrToInt(), IntToStr(), RealToStr(), CharToStr()

Esempio
```
Int Letter = StrToChar( InputField, i);
```

9.17.13 StrToInt

Descrizione
Converte un numero sotto forma di stringa in un valore numerico intero.

Sintassi
Int StrToInt(String Value)

Parametri

Value numero sotto forma di stringa

Valore resituito
il valore in formato intero

Funzioni inerenti
StrToReal(), IntToStr(), RealToStr(), CharToStr()

Esempio
```
Spessore = StrToInt(InputDialog("Spessore", "Macchina 1","10"));
```

9.17.14 StrToReal

Descrizione
Converte un numero reale sotto forma di stringa in un valore numerico reale.

Sintassi
Real StrToReal(String Value)

Parametri
Value numero sotto forma di stringa

Valore restituito
il valore in formato numerico reale

Funzioni inerenti
RealToStr(), StrToInt(), CharToStr(), IntToStr()

Esempio
```
Distanza = StrToReal(DistanzaInserita);
```

9.18 Template

9.18.1 TPageClose

Descrizione:
Chiude una pagina template specificata da un handle

Sintassi:
Void TPageClose(int Handle)

Parametri:
Handle identificativo numerico della pagina da chiudere

Valore restituito:
(nessuno)

Funzioni inerenti:
TPageOpen

Esempio:
```
int genPage;
...
genPage = TPageOpen("Genesis");
MessageBox("Click to close","Genesis page");
TPageClose(genPage);
```

9.18.2 TPageCloseByName

Descrizione:
Chiude tutti i template specificati dal nome indicato

Sintassi:
Void TPageCloseByName(string TemplateName)

Parametri:
TemplateName nome associato ai template da chiudere

Valore restituito:
(nessuno)

Funzioni inerenti:
TPageOpen

Esempio:
```
TPageOpen("Main");
MessageBox("Click to close","Main page");
TPageCloseByName("Main");
```

9.18.3 TPageOpen

Descrizione
Apre una pagina template specificata dal nome.

Sintassi
int TPageOpen(String Name)

Parametri
Name nome della pagina da aprire

Valore restituito
numero di pagina >1 (se la pagine viene aperta con successo)
0 (in caso contrario)

Funzioni inerenti
TPageClose,TPageCloseByName()

Esempio
```
int page;
page = TPageOpen("Progetto Genesis");
```

9.18.4 TPagePrint

Descrizione
Stampa la pagina template

Sintassi
Bool TPagePrint(Bool ViewPrintDlg)

Parametri
ViewPrintDlg specifica se mostrare la finestra con le preferenze di stampa prima di effettuarla
true (mostra la finestra)

Valore restituito

true (in caso l'operazione sia avvenuta con successo)
false (in caso contrario)

Funzioni inerenti
-

Esempio
```
if (TPagePrint(true)) then
      MessageBox("Stampa avvenuta","Risultati stampa");
else
      MessageBox("Stampa fallita","Risultati stampa");
end
```

9.18.5 TemplateAlarmsStatus

Descrizione
Apre il template con lo stato degli allarmi.

Sintassi
int TemplateAlarmsStatus()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateAlarmsStatus();
```

9.18.6 TemplateChart

Descrizione
Apre il template per la gestione dei grafici.

Sintassi
int TemplateChart()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() ,

TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateChart();
```

9.18.7 TemplateDefineAccess

Descrizione
Apre il template per la definizione accesso pagine.

Sintassi
int TemplateDefinePagesAccess()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateDefinePagesAccess();
```

9.18.8 TemplateDefineGroupNames

Descrizione
Apre il template per la definizione dei nomi dei gruppi.

Sintassi
int TemplateDefineGroupNames()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateDefineGroupNames();
```

9.18.9 TemplateDefineUsers

Descrizione
Apre il template per la definizione dei codici operatore.

Sintassi
int TemplateDefineUsers()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateDefineUsers();
```

9.18.10 TemplateDevicesStatus

Descrizione
Apre il template con lo stato dei dispositivi.

Sintassi
intTemplateDevicesStatus()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateDevicesStatus();
```

9.18.11 TemplateEventsStatus

Descrizione
Apre il template con lo stato degli eventi.

Sintassi
Void TemplateEventsStatus()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateEventsStatus();
```

9.18.12 TemplateGatesStatus

Descrizione
Apre il template con lo stato delle porte.

Sintassi
int TemplateGatesStatus()

Parametri
-

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateGatesStatus();
```

9.18.13 TemplateHistAlarms

Descrizione
Apre il template con lo storico allarmi.

Sintassi
int TemplateHistAlarms()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateHistAlarms();
```

9.18.14 TemplateHistEvents

Descrizione
Apre il template con lo storico eventi..

Sintassi
int TemplateHistEvents()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateHistEvents();
```

9.18.15 TemplateMakeReport

Descrizione
Apre il template per la creazione dei report.

Sintassi
int TemplateMakeReport()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateMakeReport();
```

9.18.16 TemplateMultilanguage

Descrizione
Apre il template per la gestione della lingua.

Sintassi
int TemplateMultilanguage()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateMultilanguage();
```

9.18.17 TemplatePassword

Descrizione
Apre il template per l'inserimento della password.

Sintassi
int TemplatePassword()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() ,

TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplatePassword();
```

9.18.18 TemplatePrinterSetup

Descrizione
Apre il template per la configurazione delle stampe.

Sintassi
int TemplatePrinterSetup()

Parametri:
-

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Valore restituito
-

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplatePrinterSetup();
```

9.18.19 TemplateRecipe

Descrizione
Apre il template per la gestione delle ricette.

Sintassi
int TemplateRecipe()

Parametri
-
Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateRecipe();
```

9.18.20 TemplateSystemStatus

Descrizione
Apre il template di stato del supervisore.

Sintassi
int TemplateSystemStatus()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateSystemStatus();
```

9.18.21 TemplateUserChanges

Descrizione
Apre il template degli interventi operatore..

Sintassi
int TemplateUserChanges()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateUserChanges();
```

9.18.22 TemplateViewReport

Descrizione
Apre il template per la visualizzazione dei report.

Sintassi

int TemplateViewReport()

Parametri
-

Valore restituito
L'handle della pagina associata (-1 se l'operazione è fallita)

N.B.: Il programma supervisore è in grado di gestire massimo otto finestre contemporaneamente, se tale numero viene superato l'operazione fallisce.

Funzioni inerenti
TemplatePrinterSetup(), TemplateRecipe() , TemplateSystemStatus() , TemplateDevicesStatus() , TemplateGatesStatus() , TemplateAlarmsStatus() , TemplateEventsStatus() , TemplateHistAlarms() , TemplateHistEvents() , TemplateUserChanges() , TemplateChart() , TemplateMultilanguage() , TemplateMakeReport() , TemplateViewReport() , TemplatePassword()

Esempio
```
TemplateViewReport();
```

9.19 Template Objects

9.19.1 Generic

9.19.1.1 TObjBeginUpdate

Descrizione
Inizializza la sessione di aggiornamento delle proprietà dell'oggetto specificato.
Quando si desidera impostare alcune proprietà dell'oggetto tramite le funzioni TObjSetPropertyBool(), TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(),TObjSetPropertyUnsigned(), è necessario inviare prima la funzione TObjBeginUpdate() per inizializzare la sessione di modifica delle proprietà.
Una volta impostate tutte le proprietà desiderate, è necessario richiamare la funzione TObjEndUpdate() per rendere effettive le modifiche.

Sintassi
Void TObjBeginUpdate(Int Id)

Paramentri
Id identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(), TObjSetPropertyUnsigned(),TObjEndUpdate()

Esempio
```
...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...
```

9.19.1.2 TObjEndUpdate

Descrizione
Termina la sessione di modifica delle proprietà dell'oggetto e le rende effettive.
Quando si desidera impostare alcune proprietà dell'oggetto tramite le funzioni TObjSetPropertyBool(), TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(),TObjSetPropertyUnsigned(), è necessario inviare prima la funzione TObjBeginUpdate() per inizializzare la sessione di modifica delle proprietà.
Una volta impostate tutte le proprietà desiderate, è necessario richiamare la funzione TObjEndUpdate() per rendere effettive le modifiche.

Sintassi
Void TObjEndUpdate(Int Id)

Paramentri
Id identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(), TObjSetPropertyUnsigned(),TObjBeginUpdate()

Esempio
```
...
TObjBeginUpdate(100);
TObjSetPropertyReal(100,"ScaleMin",20);
TObjSetPropertyReal(100,"ScaleMax",80);
TObjEndUpdate(100);
...
```

9.19.1.3 TObjFunction

Descrizione
Invia un comando all'oggetto specificato.
Per esempio, sull'oggetto Chart esiste la possibilità di aprire alcune finestre di configurazione (come la configurazione del gruppo grafici o dell'intervallo di visualizzazione) o la possibilità di simulare la pressione dei tasti "Avanti","Indietro" e "Reset Zoom".
In questo modo possono essere creati dei pulsanti personalizzati (tramite oggetti Button, Label e Bitmap) per manipolare l'oggetto Chart.

Sintassi
Void TObjFunction(Int Id,Int Function)

Paramentri
Id identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
Function comando da eseguire. Consultare l'help dell'oggetto nel Template Builder per verificare quali comandi sono supportati

Valore restituito

-

Funzioni inerenti

Esempio

```
...
TObjFunction(1,2);
...
```

9.19.1.4 TObjGetH

Descrizione
Restituisce la dimensione verticale dell'oggetto specificato.

Sintassi
Int TObjGetH(Int Id)

Parametri
Id identificatore dell'oggetto

Valore restituito
Altezza dell'oggetto

Funzioni inerenti
TObjGetW(), TObjSetSize()

Esempio

```
ObjHeight=TObjGetH(ObjID);
```

9.19.1.5 TObjGetLButtonDownXY

Descrizione
Restituisce le coordinate X,Y del puntatore del mouse al momento dell'ultima pressione del tasto sinistro, all'interno dell'oggetto specificato dall'identificatore.

Sintassi
Int TObjGetLButtonDownXY(Int Id)

Paramentri
Id identificatore dell'oggetto

Valore restituito
Valore contenente le coordinate XY
Coordinata X=Valore & 0x0000FFFF
Coordinata Y=Valore / 65536

Funzioni inerenti
TObjGetLButtonUpXY()

Esempio

```
int LButtonDnXY=TobjGetLButtonDownXY(2);
int LButtonDnX=BitAnd(LButtonDnXY,65535);
int LButtonDnY=LButtonDnXY/65536;
```

9.19.1.6 TObjGetLButtonUpXY

Descrizione
Restituisce le coordinate X,Y del puntatore del mouse al momento dell'ultimo rilascio del tasto sinistro, all'interno dell'oggetto specificato dall'identificatore.

Sintassi
Int TObjGetLButtonUpXY(Int Id)

Paramentri
Id identificatore dell'oggetto

Valore restituito
Valore contenente le coordinate XY
Coordinata X=Valore & 0x0000FFFF
Coordinata Y=Valore / 65536

Funzioni inerenti
TObjGetLButtonDownXY()

Esempio
```
int LButtonUpXY=TobjGetLButtonUpXY(2);
int LButtonUpX=BitAnd(LButtonUpXY,65535);
int LButtonUpY=LButtonUpXY/65536;
```

9.19.1.7 TObjGetPropertyInt

Descrizione
Restituisce una proprietà di tipo Integer dell'oggetto specificato tramite identificatore.

Sintassi
Int TObjGetPropertyInt(Int Id, string PropertyName)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da leggere. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà che possono essere lette tramite questa funzione.

Valore restituito
Valore della proprietà letta.

Funzioni inerenti
-

Esempio
```
...
int Index;
Index=TObjGetPropertyInt(100,"ItemSelected");
...
```

9.19.1.8 TObjGetPropertyString

Descrizione
Restituisce una proprietà di tipo stringa dell'oggetto specificato tramite identificatore.
Se la proprietà non esiste allora ritorna stringa vuota.

Sintassi
String TObjGetPropertyString(Int Id, string PropertyName)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da leggere. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà che possono essere lette tramite questa funzione.

Valore restituito
Valore della proprietà letta.

Funzioni inerenti
-

Esempio
```
...
string Path = TObjGetPropertyString(100,"PathSelected");
...
```

9.19.1.9 TObjGetStatus

Descrizione
Restituisce lo stato dell'oggetto specificato.

Sintassi
Int TObjGetStatus(Int Id)

Parametri
Id identificatore dell'oggetto

Valore restituito
stato dell'oggetto

Funzioni inerenti
TObjSetStatus()

Esempio
```
CurrStatus=TObjGetStatus(CurrObject);
```

9.19.1.10 TObjGetText

Descrizione
Restituisce il testo dell'oggetto specificato, ovviamente questa funzione non ha senso per quegli oggetti che non sono dotati di testo (es: BitMap).

Sintassi
String TObjGetText(Int Id)

Parametri
Id identificatore dell'oggetto

Valore restituito
testo dell'oggetto

Funzioni inerenti
TObjSetText()

Esempio
```
Caption=TobjGetText(CurrObj);
```

9.19.1.11 TObjGetX

Descrizione
Restituisce l'ascissa dell'oggetto specificato tramite identificatore.

Sintassi
Int TObjGetX(Int Id)

Paramentri
Id identificatore dell'oggetto

Valore restituito
l'ascissa del'oggetto specificato

Funzioni inerenti
TObjSetXY(),TObjGetY()

Esempio
```
posX=TObjGetX(CurrObj);
```

9.19.1.12 TObjGetY

Descrizione
Restituisce l'ordinata dell'oggetto specificato tramite identificatore.

Sintassi
Int TObjGetY(Int Id)

Paramentri
Id identificatore dell'oggetto

Valore restituito
l'ordinata del'oggetto specificato

Funzioni inerenti
TObjSetXY(),TObjGetX()

Esempio
```
posY=TObjGetY(CurrObj);
```

9.19.1.13 TObjGetW

Descrizione
Restituisce la dimensione orizzontale dell'oggetto specificato.

Sintassi
Int TObjGetW(Int Id)

Parametri
Id identificatore dell'oggetto

Valore restituito
Larghezza dell'oggetto

Funzioni inerenti
TObjGetH(), TObjSetSize()

Esempio
```
ObjWidth=TObjGetW(ObjID);
```

9.19.1.14 TObjSetPropertyBool

Descrizione
Modifica una proprità di tipo Bool dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetPropertyBool(Int Id, string PropertyName,bool Value)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da impostare. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà impostabili tramite questa funzione.
Value	valore da impostare

Valore restituito
-

Funzioni inerenti
TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(),TObjSetPropertyUnsigned(), TObjBeginUpdate(),TObjEndUpdate()

Esempio
```
...
TObjBeginUpdate(100);
TObjSetPropertyBool(100,PropertyName,true);
TObjEndUpdate(100);
...
```

9.19.1.15 TObjSetPropertyInt

Descrizione
Modifica una proprità di tipo Integer dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetPropertyInt(Int Id, string PropertyName,int Value)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da impostare. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà impostabili tramite questa funzione.
Value	valore da impostare

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyReal(),TObjSetPropertyString(),TObjSetPropertyUnsigned(),

TObjBeginUpdate(),TObjEndUpdate()

Esempio

```
...
TObjBeginUpdate(100);
TObjSetPropertyInt(100,"DecimalNumber",1);
TObjEndUpdate(100);
...
```

9.19.1.16 TObjSetPropertyReal

Descrizione
Modifica una proprità di tipo Real dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetPropertyReal(Int Id, string PropertyName,real Value)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da impostare. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà impostabili tramite questa funzione.
Value	valore da impostare

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyInt(),TObjSetPropertyString(),TObjSetPropertyUnsigned(), TObjBeginUpdate(),TObjEndUpdate()

Esempio

```
...
TObjBeginUpdate(100);
TObjSetPropertyString(100,"DescriptionText","Hello World");
TObjEndUpdate(100);
...
```

9.19.1.17 TObjSetPropertyString

Descrizione
Modifica una proprità di tipo String dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetPropertyStringl(Int Id, string PropertyName,string Value)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da impostare. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà impostabili tramite questa funzione.
Value	valore da impostare

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyUnsigned(), TObjBeginUpdate(),TObjEndUpdate()

Esempio
```
...
TObjBeginUpdate(100);
TObjSetPropertyString(100,"DescriptionText","Hello World");
TObjEndUpdate(100);
...
```

9.19.1.18 TObjSetPropertyUnsigned

Descrizione
Modifica una proprità di tipo Unsigned Integer dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetPropertyUnsigned(Int Id, string PropertyName,unsigned int Value)

Paramentri

Id	identificatore dell'oggetto. E' il numero impostato nella proprietà "ID" del' oggetto tramite Template Builder.
PropertyName	nome della proprietà da impostare. Consultare l'help dell'oggetto nel Template Builder per conoscere l'elenco delle proprietà impostabili tramite questa funzione.
Value	valore da impostare

Valore restituito
-

Funzioni inerenti
TObjSetPropertyBool(),TObjSetPropertyInt(),TObjSetPropertyReal(),TObjSetPropertyString(), TObjBeginUpdate(),TObjEndUpdate()

Esempio
```
...
TObjBeginUpdate(100);
TObjSetPropertyUnsigned(100,PropertyName,1);
TObjEndUpdate(100);
...
```

9.19.1.19 TObjSetSize

Descrizione
Modifica le dimensioni dell'oggetto specificato tramite identificatore.

Sintassi
Void TObjSetSize(Int Id, Int w, Int h)

Paramentri

Id	identificatore dell'oggetto
w	larghezza (width)
h	altezza (height)

Valore restituito
-

Funzioni inerenti
TObjGetW(),TObjGetH()

Esempio
```
TObjSetSize(CurrObj,50,50);
```

9.19.1.20 TObjSetStatus

Descrizione
Modifica lo stato dell'oggetto specificato.

Sintassi
Void TObjSetStatus(Int Id, Int status)

Parametri
Id identificatore dell'oggetto
Status nuovo stato

Valore restituito
-

Funzioni inerenti
TObjGetStatus()

Esempio
```
TObjSetStatus(ObjNotIdentified,0);
```

9.19.1.21 TObjSetText

Descrizione
Modifica il testo dell'oggetto specificato, ovviamente questa funzione non ha senso per quegli oggetti che non sono dotati di testo (es: BitMap).

Sintassi
Void TObjSetText(Int Id, String text)

Parametri
Id identificatore dell'oggetto
Text nuovo testo

Valore restituito
-

Funzioni inerenti
TObjGetText()

Esempio
```
TObjSetText(NuovoObj,"Nuovo");
```

9.19.1.22 TObjSetXY

Descrizione
Modifica la posizione dell'oggetto specificato tramite il suo identificatore.

Sintassi
Void TObjSetXY(Int Id, Int x, Int y)

Paramentri

Id identificatore dell'oggetto
x nuova ascissa
y nuova ordinata

Valore restituito
-

Funzioni inerenti
TObjGetX(),TObjGetY()

Esempio
```
TObjSetXY(CurrObj, 100, 100);
```

9.19.2 HistAlarmsView

9.19.2.1 HisViewEnablePrintOnCreation

Descrizione
Stampa in automatico lo storico allarmi o eventi subito dopo l'apertura del template che contiene l'oggetto HisView.

Sintassi
void HisViewEnablePrintOnCreation()

Parametri
-

Valore restituito
-

Funzioni inerenti
-

Esempio
```
HisViewEnablePrintOnCreation();
```

9.19.2.2 HisViewSetTimeRange

Descrizione
Imposta l'intervallo temporale di un oggetto *historical view* contenuto in un template.
L'oggetto chart deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi
void HisViewSetTimeRange(
int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int eDay, int eMonth, int eYear, int eHour, int eMin, int eSec)

Parametri
sDay giorno d'inizio
sMonth mese d'inizio
sYear anno d'inizio
sHour ora d'inizio
sMin minuti d'inizio
sSec secondi d'inizio
eDay ora di fine
eMonth mese di fine
eYear anno di fine

eHour ora di fine
eMin minuti di fine
eSec secondi di fine

Valore restituito
-

Funzioni inerenti
HisViewSetTimeRangeStartWidth(),HisViewSetTimeRangeEndWidth()

Esempio
```
HisViewSetTimeRange (24,1,1975,0,0,0 ,24,1,2000,0,0,0);
```

9.19.2.3 HisViewSetTimeRangeStartWidth

Descrizione
Imposta la **data ora di inizio** e la **finestra temporale** di visualizzazione di un oggetto HisView contenuto in un template.
L'oggetto deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi
void HisViewSetTimeRangeStartWidth(
int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int eDay, int Width)

Parametri
sDay giorno d'inizio
sMonth mese d'inizio
sYear anno d'inizio
sHour ora d'inizio
sMin minuti d'inizio
sSec secondi d'inizio
Width finestra temporale (secondi)

Valore restituito
-

Funzioni inerenti
HisViewSetTimeRange(),HisViewSetTimeRangeEndWidth()

Esempio
```
HisViewSetTimeRangeStartWidth (24,1,1975,0,0,0 ,3600);
```

9.19.2.4 HisViewSetTimeRangeEndWidth

Descrizione
Imposta la **data ora di fine** e la **finestra temporale** di visualizzazione di un oggetto HisView contenuto in un template.
L'oggetto deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi
void HisViewSetTimeRangeEndWidth(
int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int eDay, int Width)

Parametri

sDay	giorno fine
sMonth	mese fine
sYear	anno fine
sHour	ora fine
sMin	minuti fine
sSec	secondi fine
Width	finestra temporale (secondi)

Valore restituito
-

Funzioni inerenti
HisViewSetTimeRange(),HisViewSetTimeRangeStartWidth()

Esempio
```
HisViewSetTimeRangeEndWidth (24,1,1975,0,0,0 ,3600);
```

9.19.3 Chart

9.19.3.1 ChartEndDrawingFlagReset

Descrizione
Resetta il flag di fine tracciamento grafico.
Quando viene richiamato un template contenente oggetto chart, la funzione ChartEndDrawingFlag viene impostato a *true* quando è stato completato il tracciamento di tutti i grafici.

Sintassi
void ChartEndDrawingFlagReset()

Parametri
-

Valore restituito
-

Funzioni inerenti
ChartEndDrawingFlagStatus()

Esempio
```
ChartEndDrawingFlagReset();
TOpenPage("ChartContainer");
while ( ChartEndDrawingFlagStatus() == false )
end
```

9.19.3.2 ChartEndDrawingFlagStatus

Descrizione
Restituisce lo stato del flag di fine tracciamento grafico.
Quando viene richiamato un template contenente oggetto chart, la funzione ChartEndDrawingFlag viene impostato a *true* quando è stato completato il tracciamento di tutti i grafici.

Sintassi
Bool ChartEndDrawingFlagStatus()

Parametri
-

Valore restituito
True (se il grafico è stato tracciato)
False (in caso contrario)

Funzioni inerenti
ChartEndDrawingFlagReset()

Esempio
```
ChartEndDrawingFlagReset();
TOpenPage("ChartContainer");
while ( ChartEndDrawingFlagStatus() == false )
end
```

9.19.3.3 ChartSetTimeRange

Descrizione
Imposta l'intervallo temporale di un oggetto chart contenuto in un template.
L'oggetto chart deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi
void ChartSetTimeRange(
int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int eDay, int eMonth, int eYear, int eHour, int eMin, int eSec)

Parametri

sDay	giorno d'inizio
sMonth	mese d'inizio
sYear	anno d'inizio
sHour	ora d'inizio
sMin	minuti d'inizio
sSec	secondi d'inizio
eDay	giorno di fine
eMonth	mese di fine
eYear	anno di fine
eHour	ora di fine
eMin	minuti di fine
eSec	secondi di fine

Valore restituito
-

Funzioni inerenti
ChartSetTimeRangeStartWidth(),ChartSetTimeRangeEndWidth()

Esempio
```
ChartSetTimeRange (24,1,1975,0,0,0 ,24,1,2000,0,0,0);
```

9.19.3.4 ChartSetTimeRangeStartWidth

Descrizione
Imposta la **data ora di inizio** e la **finestra temporale** di visualizzazione di un oggetto chart contenuto in un template.
L'oggetto chart deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi
void ChartSetTimeRangeStartWidth(int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int

Width)

Parametri

sDay	giorno d'inizio
sMonth	mese d'inizio
sYear	anno d'inizio
sHour	ora d'inizio
sMin	minuti d'inizio
sSec	secondi d'inizio
Width	finestra temporale (secondi)

Valore restituito

-

Funzioni inerenti

ChartSetTimeRange(),ChartSetTimeRangeEndWidth()

Esempio

```
ChartSetTimeRangeStartWidth (24,1,1975,0,0,0 ,3600);
```

9.19.3.5 ChartSetTimeRangeEndWidth

Descrizione

Imposta la **data ora di fine** e la **finestra temporale** di visualizzazione di un oggetto chart contenuto in un template.
L'oggetto chart deve avere impostata la proprietà *TimeRange* come *esterno, inizio esterno,* o *fine esterno.*

Sintassi

void ChartSetTimeRangeEndWidth(int sDay, int sMonth, int sYear, int sHour, int sMin, int sSec, int Width)

Parametri

sDay	giorno fine
sMonth	mese fine
sYear	anno fine
sHour	ora fine
sMin	minuti fine
sSec	secondi fine
Width	finestra temporale (secondi)

Valore restituito

-

Funzioni inerenti

ChartSetTimeRange(),ChartSetTimeRangeStartWidth()

Esempio

```
ChartSetTimeRangeEndWidth (24,1,1975,0,0,0 ,3600);
```

10 Elementi del linguaggio

10.1 Introduzione

Il linguaggio è composto da elementi di base (lettere, numeri, simboli di punteggiature, etc) da regole semantiche e da una sintassi che specificano la correttezza dei comandi che si possono creare.
Le istruzioni non sono altro che espressioni che coinvolgono delle variabili e delle funzioni tramite operatori forniti dal linguaggio stesso.
Le variabili sono zone di memoria dove è possibile immagazzinare i dati necessari, chiamate, appunto variabili, perché possono assumere differenti valori.
Esistono diverse categorie o tipi di variabili che differiscono in base all'informazione che rappresentano e allo spazio che occupano in memoria.
Le funzioni sono invece insiemi di istruzioni che vengo raggruppate per comodità, per evitare di doverle riscrivere e perché in questa maniera rappresentano un specie di istruzione più complessa.
Così è possibile 'arricchire' il linguaggio di una sorta di nuove istruzioni.
Gli operatori sono gli stessi che si ritrovano anche nella matematica elementare con l'aggiunta di altri nuovi.
Per semplificare la scrittura del codice in un determinato linguaggio esistono strutture come condizioni, cicli, cicli condizionati ecc.
Le condizioni sono blocchi di istruzioni eseguite soltanto se si verifica la condizione.
I cicli sono blocchi di istruzione eseguiti un numero di volte definito.
I cicli condizionati sono blocchi di istruzioni eseguiti finché non viene verificata la condizione.
Le espressioni permettono invece di costruire elementi complessi a partire da elementi più semplici.
Gli operatori che permettono di legare insieme le espressioni.
Tutte queste istruzioni vengono scritte in files di testo che poi l'interprete decifra ed eseguire.

10.2 Premesse

I files del linguaggio hanno come estensione WLL (es: Filatoio2.wll).
Per l'interprete del linguaggio non esiste differenza tra maiuscole e minuscole (in questo caso si parla di *case insesitive*):

```
Real Valore;    // dichiarazione della variabile reale 'valore' (la doppia barra // specifica l'inizio di un commento)
Real valore;
```

L'interprete leggendo questo frammento di codice genererà un errore perché riscontrerà che l'elemento *valore* è stato definito due volte.
Si vedrà meglio successivamente cosa significa 'definire' e come 'elemento' sia soltanto un termine generico che nei casi specifici assume i valori 'variabile', 'funzione', 'costante' ecc.
È importante separare gli elementi con spazi o altro, altrimenti questi verranno riconosciuti come entità singole.

```
realValore;                    // se si voleva dichiare la variabile Valore si è commesso un errore!!!
                                    // l'interprete vede solo un elemento ovvero 'realvalore'
```

In ogni file possono essere definite funzioni e variabili globali visibili a tutte le funzioni definite.
Le variabili globali vengono definite all'inizio del file.
La visibilità è la possibilità di accedere ad un dato, se una variabile non è visibile allora non la si può utilizzare in nessuna maniera.

10.3 Variabili

Sono elementi che contengono i valori da manipolare durante l'esecuzione di un programma o di una funzione.

Il linguaggio mette a disposizione due tipi di variabili:

- **Globali** e quindi visibili a tutte le funzioni del programma anche se queste si trovano in file differenti (visibilità globale)

Sintassi
Global *Type Name [= Default value]*

Type è il tipo della variabile.
Name è il nome che si vuole attribuire alla variabile.

Esempio
```
Global Real TemperaturaEsatta;
Global Int ZeroAssoluto=273;
```

- **Locali** ovvero definite all'interno di ogni funzione e quindi utilizzabili soltanto nel contesto della funzione (visibilità locale)

Sintassi
Type Name [= Default value]

Type è il tipo della variabile.
Name è il nome che si vuole attribuire alla variabile.

Esempio
```
Function Int TemperaturaAssoluta( integer T0 )
   Real TemperaturaPT01;
  ...
End
```

La variabile TemperaturaPT01 è locale alla funzione TemperaturaAssoluta e quindi non utilizzabile al di fuori di essa, ma nessuno vieta di utilizzare lo stesso nome per un altra variabile locale in un'altra funzione.

10.4 Tipi

10.4.1 Introduzione

Le variabili contengono informazioni diverse e a seconda dell'informazione che contengono vengono divise in categorie o tipi; Esistono diversi tipi di variabile.

Int	valori interi
Unsigned	valori interi senza segno
Real	valori reali
Bool	valori booleani
String	stringhe

10.4.2 Int

Sono valori interi a 32 bit dotati di segno e possono assumere valori da -2147483648 a +2147483647 : valori al di fuori di questi limiti verranno troncati ai primi 32 bit meno significativi.

Esempio
```
Int Counter = -2147483648;
Int Difference = +2147483647;
```

10.4.3 Unsigned

Sono valori interi a 32 bit senza segno e possono assumere valori da 0 a 4294967295 : valori al di fuori di questi limiti verrano troncati a primi 32 bit meno significativi.

Esempio
```
Int Counter = 0;
Int Difference = 4294967295;
```

10.4.4 Real

Sono numeri nel formato a virgola mobile.

Esempio
```
Real Pi = 3.141592;
```

10.4.5 Bool

Sono valori booleani quindi possono assumere solo due stati **true** (vero) oppure **false** (falso).

Esempio
```
Bool LedOn = true;
```

10.4.6 String

Sono stringhe di caratteri di lunghezza variabile.

Esempio
```
String Nome="Settembre";
```

10.5 Funzioni

La funzione è una procedura cioè un elenco di istruzioni scritte dallo sviluppatore dell'applicazione atte a svolgere un compito specifico come per esempio calcolare il risultato di una formula matematica, leggere o scrivere un file da disco, visualizzare dati in un certo formato, inviare comandi ai dispositivi connessi con l'applicazione, generare rapporti di produzione ed innumerevoli altre operazioni.
Le funzioni sono quindi delle macroistruzioni create dallo sviluppatore stesso in base alle proprie esigenze.Un vantaggio dell'utilizzo delle funzioni è quello di ridurre la necessità di scrivere righe di codice ridondante: se devo calcolare l'area del rettangolo per 10 rettangoli diversi fra loro, non è necessario riscrivere per 10 volte la formula dell''area del rettangolo, bensì posso definire una Funzione che riceverà in ingresso due parametri cioè Base ed Altezza e ritornerà come risultato il calcolo dell'area del rettangolo: la formula per tale calcolo verrà scritta una sola volta nella funzione: a questo punto per calcolare l'area dei 10 rettangoli mi basta semplicemente chiamare 10 volte la funzione appena definita passandogli volta per volta i parametri relativi al rettangolo di cui volgio calcolarne l'area.

Una funzione è caratterizzata da:
un valore di ritorno che può essere o uno dei tipi di variabile supportati dal software (tipi) oppure *void* se la funzione non deve restituire nulla.
un nome unico (senza distinzione fra maiuscolo e minuascolo) che identifica la funzione in tutto il programma.
una lista di parametri (opzionale) che sono i valori di cui la funzione necessita per svolgere il proprio compito.
direttive (opzionali) che sono comandi particolari che arricchiscono le funzioni.
variabili locali (opzionali) necessarie allo svolgimento della funzione.
una sequenza di istruzioni che caratterizza la funzione.

È consigliabile dare nomi significativi alle funzioni per semplificare il lavoro, quindi se una funzione svolge il calcolo della temperatura in gradi Kelvin allora la si potrà chiamare 'CalcolaTemperaturaK'.

[ ] le parentesi quadre indicano che l'elemento racchiuso in esse non è strettamente necessario.

Sintassi
Function *Type Name([Parameter List])*
  [Directives]
  [Local Vars]
  Instructions
End

Type è il tipo di informazione che restituisce la funzione (tipi).
Name è il nome che identifica la funzione.
Parameter List sono la lista ed il tipo di dati da fornire all funzione perché possa svolgere il proprio compito.
Local Vars è la definizione di tutte le variabili necessarie allo svolgimento.
Instrucitions è la sequenza di istruzioni che la funzione esegue.

Le direttive indicano:
#Macro: la funzione risulterà richiamabile dall'utente attraverso il menù **macro**.
#Startup: la funzione viene eseguita all'avvio del supervisore in background. È quindi possibile generare funzioni cicliche che eseguono controlli o azioni dettate da certe condizioni.
#Shutdown: la funzione viene eseguita alla chiusura del supervisore.
#Modal: Il supervisore aspetta la fine dell'esecuzione della funzione (questa modalità impedisce qualsiasi operazione da parte dell'operatore ma non interrompe il campionamento e la registrazione delle porte).

Esempio

```
Function Void MainCycle()
#Startup
        int in1;
        int in2;

        while (WindowIsOpen())
          in1 = GetDigGateValue("101IN",1);
          in2 = GetDigGateValue("101IN",2);

          if (in1+in2 == 2) then
                  SetDigGateValue("103OUT",1,1);
          else
                  SetDigGateValue("103OUT",1,0);
          end

          in1 = GetDigGateValue("101IN",48);
          in2 = GetDigGateValue("101IN",49);

          if (in1+in2 > 0) then
                  SetDigGateValue("103OUT",2,1);
          else
                  SetDigGateValue("103OUT",2,0);
          end

          if (Abs(GetCmpGateValue("Tr-Ts",1)) > GetNumGateValue("102DeltaT",1)) then
```

```
            SetDigGateValue("103OUT",3,1);
        else
            SetDigGateValue("103OUT",3,0);
        end

        Sleep(100);
    end
end
```

Incolonnare correttamente le istruzioni permette una migliore lettura e comprensione di una funzione.
È possibile richiamare funzioni all'apertura ed alla chiusura di un *Template* oppure tramite dei comandi (Es. un *Button*) definiti nel *Template* stesso.

N.B. Non è possibile eseguire una funzione se essa è già in esecuzione.

10.6 Istruzioni

10.6.1 Premesse

Ogni istruzione deve terminare con il ';'.
Le parole riservate **end, if, while** etc non hanno bisogno di essere seguite dal ';'.

10.6.2 Chiamate di funzione

Si utilizza quando si vuole richiamare una funzione.

Sintassi
FunctionName(*[Parameter Expressions]*);

Esempio
```
PompaSpenta = PompaAttiva() - 3;
AttivaPompaNum(3);
```

10.6.3 Assegnamento

Quando si vuole attribuire un valore ad una variabile si utilizza l'assegnamento.

Sintassi
VarName = *Expression*;

Esempio
```
NumeroDiPID = (( 9 * Var1 ) / ( 1 - Var3 )) * ( 2 - Var2 );
Temp = TriggerValue - CurrValue;
```

10.6.4 Valore di ritorno

Return si utilizza per uscire dalla funzione corrente ritornando a quella chiamante, eventualmente restituendo un valore.

Sintassi
Return *Expression*;

Esempio
```
Function void Caller()
   ...
   Int A;
   ...
```

```
   ASqr = Quadrato(A);
   ...
End


Function int Quadrato(int Numero)
   int Quad;
   Quad = Numero * Numero;
   Return (Quad);
End
```

10.6.5 If Then Else

Se viene utilizzato nella forma:
If () Then
...
End

esegue il blocco di istruzioni comprese fra *Then* e *End* se la condizione nell'istruzione *If()* risulta verificata.

Se viene utilizzato nella forma:
If () Then
...
Else
...
End

esegue il blocco di istruzioni comprese fra *Then* e *Else* se la condizione specificata nell'istruzione *If()* risulta verificata, altrimenti esegue il blocco di istruzioni specificate fra *Else* ed *End*.

Sintassi
If (*Condition*) Then *Instructions*
*[*Else *Instructions]*
End

Esempio:
```
If (ValvolaAperta == 1) Then
   ChiudiValvolaPrimaria();
Else
   ChiudiValvolaNum(ValvolaAperta);
End

If (Temperatura < 75) Then
   Giri = 2000 - Temperatura * 5;
   SelezionaGiriMotoreNum(5, Giri);
End
```

10.6.6 Ciclo For

Ripete per un numero definito di volte il blocco di istruzioni comprese fra *For* e *End*.
Necessita di una variabile intera come contatore.

Sintassi
For *VarName* = *Expression* To *Expression* Do
  [Instructions]
End

Esempio
```
For NumPompa = 2 To (15-PompeSpente()) Do
   SelezionePompa(NumPompa);
   Livello = ControllaLivelloLiquido();
   AttivaPompa(Livello / 10);
End
```

Note:
Il codice viene eseguito con un'alta priorità all'interno dell'applicazione; costruire un loop di lunga durata (come può essere fatto con l'istruzione "For") può causare il rallentamento del tempo di risposta dell'applicazione: per evitare ciò è consigliato l'inserimento di una istruzione "Sleep()" all'interno del loop in modo da consentire al sistema di gestire il resto dell'applicazione (per esempio il refresh delle finestre,ecc.) più efficientemente e con maggiore fluidità .

10.6.7 Ciclo While

Se la condizione specificata nelle parentesi del While() risulta essere verificata, esegue il blocco di istruzioni comprese fra While() ed end, dopodichè ritorna a testare la condizione nel While() e ripete il loop fino a quando la condizione non risulterà più verificata.

Sintassi
While (*Comparison*)
  [Instructions]
End

Esempio
```
While (Temperatura < 100)
   Velocita = Velocita + 1;
   LeggiTemperatura(Temperatura);
   Sleep(100);
End
```

Note:
Il codice viene eseguito con un'alta priorità all'interno dell'applicazione; costruire un loop di lunga durata o infinito (come può essere fatto con l'istruzione "While") può causare il rallentamento del tempo di risposta dell'applicazione: per evitare ciò è consigliato l'inserimento di una istruzione "Sleep()" all'interno del loop in modo da consentire al sistema di gestire il resto dell'applicazione (per esempio il refresh delle finestre,ecc.) più efficientemente e con maggiore fluidità .

10.6.8 Ciclo Do While

Esegue il blocco di istruzioni compreso fra *Do* e *While()* almeno una volta, e se la condizione specificata nelle parentesi del *While()* risulta essere verificata, ritorna al *Do* e ripete di nuovo l'operazione.
Rimane in questo loop fino a quanto la condizione nel *While()* non risulterà più verificata.

Sintassi
Do
  [Instructions]
While (*Comparison*)

Esempio

```
i = 0;
Do
   LightOn(i);
   i = i + 1;
While( i < 10)
```

Note:
Il codice viene eseguito con un'alta priorità all'interno dell'applicazione; costruire un loop di lunga durata o infinito (come può essere fatto con l'istruzione "Do - While") può causare il rallentamento del tempo di risposta dell'applicazione: per evitare ciò è consigliato l'inserimento di una istruzione "Sleep()" all'interno del loop in modo da consentire al sistema di gestire il resto dell'applicazione (per esempio il refresh delle finestre,ecc.) più efficientemente e con maggiore fluidità .

10.7 Espressioni

Le espressioni sono gli elementi su cui si basa il linguaggio.
Un'espressione puo' essere una costante, una variabile oppure una equazione complessa quindi non viene eseguita un'azione ma viene semplicemente valutata.
Come nelle espressioni matematiche gli elementi sono legati da operatori logici e matematici.
Il risultato di un'espressione è un valore che a sua volta può essere utilizzato come elemento di un'altra espressione; in questa maniera è posssibile articolare formule anche molto complesse.
Alcuni esempi di espressione:

ESEMPI DI ESPRESSIONE	DESCRIZIONE
3	una costante numerica
ValorePortaA	una variabile
log(14)	un funzione che restituisce un valore
4 + ValorePortaB	compare l'operatore di somma
PortaAon && PortaBon	compare l'operatore boolean AND
Alfa + 2 * exp(Beta - Gamma - D)	espressione complessa

Valutando le espressioni si ottiene un valore che può essere utilizzato come parametro da assegnare ad una variabile o ad una funzione, oppure testato in una condizione.
Ecco un esempio di utilizzo di espressioni estrapolato da una procedura:

```
...
A = 3;
B = ValorePortaA;
PartenzaScala (log(14), 4 + ValorePortaB);
if (PortaAon && PortaBon) then
  AttivaAllarme();
end
Delta = Delta * 2 + Alfa + 2 * exp(Beta - Gamma - D);
...
```

Gli operatori impiegati all'interno delle equazioni hanno, come in matematica, una priorità differente; quindi per rendere più chiara e versatile la scrittura delle equazioni è possibile utilizzare le parentesi tonde.

```
ValoreMedio = (PortaA + PortaB + PortaC) / 3;
```

Si faccia attenzione a non combinare espressioni logiche con espressioni aritmetiche.
Le espressioni logiche si trovano spesso dove è necessario un test su una condizione, come nei cicli

while e nelle condizioni if..then.

```
Esempio di espressione corretta:
if ( (A && (D == 2.50)) || (B < 10) )  then
 ...
 ...
end
```

```
Esempio di espressione NON corretta:
bool PulsanteAttivo();
...
A = ValoreMedio * 3.14 + PulsanteAttivo();
```

L'espressione precedente è errata in quanto PulsanteAttivo() restituisce un valore logico che non puo' essere combinato con uno aritmetico.

Note:
Se un'espressione è formata da un operatore booleano (&& o ||) e da due o più sottoespressioni operanti su variabili di tipo non omogeneo fra di loro è necessario l'utilizzo delle parentesi () per ogni sottoespressione.

Esempio di espressione corretta;

```
...
int A = 1;
int B = 5;
string S1 = "900";
string S2 = "1000";
if (( A > B )  &&  ( S1 > S2 )) then
...
end
```

Esempio di espressione NON corretta;

```
...
int A = 1;
int B = 5;
string S1 = "900";
string S2 = "1000";
if ( A > B && S1 > S2 ) then
...
end
```

10.8 Operatori logici

Gli operatori sono elementi del linguaggio che congiungono le espressioni permettendo cosi di costruirne di più sofisticate.
E' un insieme contenente operatori per la logica e operatori per l'algebra.

OPERATORE	FUNZIONE
Generici:	
()	priorità
Logici	
==	uguaglianza
!=	disuguaglianza

>=	maggioranza o uguaglianza
<=	minoranza o uguaglianza
>	maggioranza
<	minoranza
&&	AND
||	OR
Algebrici	
+	somma
-	sottrazione
*****	prodotto
/	divisione

Note:
- Per l'elenco delle altre funzioni matematiche disponibili, consultare il capitolo API(interfaccia di programma) - Math .
- E' possibile utilizzare l'operatore somma anche con le stringhe, ottenendone così il concatenamento.