www.asix.com.pl

Asix.Evo - Wyrażenia i funkcje

Dok. Nr PLP7E010
Wersja: 2014-03-25

ASKOM

ASKOM Sp. z o. o., ul. Józefa Sowińskiego 13, 44-121 Gliwice,
tel. +48 32 3018100, fax +48 32 3018101,

http://www.askom.com.pl, e-mail: biuro@askom.com.pl

Spis treści

1 Użycie wyrażeń i funkcji

Podstawowym mechanizmem stosowanym w parametryzacji aplikacji Asix.Evo są wyrażenia. Pozwalają one na dynamiczne wyliczanie wartości w trakcie pracy aplikacji w zależności od aktualnego stanu różnych elementów systemu, w szczególności zmiennych procesowych.

Wyrażenia są używane między innymi w:

- Definicjach właściwości obiektów i elementów menu
- Definicjach właściwości globalnych
- Argumentach akcji operatorskich i funkcji
- Funkcjach przeliczających zmiennych procesowych
- Parametrach wykrywania alarmów

2 Składnia wyrażeń

2.1 Elementy wyrażeń

Konstrukcja wyrażeń Asix.Evo jest zbliżona do składni wyrażeń arytmetycznych języka C. Wyrażenia zbudowane są z operatorów określających wykonywane działania oraz elementów wyrażeń będących argumentami tych działań. Możliwe są poniższe rodzaje elementów, opisane w kolejnych podrozdziałach.

2.1.1 Liczby

Liczby całkowite, zmiennoprzecinkowe lub szesnastkowe. Liczby zmiennoprzecinkowe zawsze używają znaku kropki jako separatora części ułamkowej. Liczby szesnastkowe zapisywane są z prefiksem *0x*.

Przykłady:

77 -14 45.34 12e2 0x0f

2.1.2 Teksty

Napisy złożone z dowolnych znaków ograniczonych znakami ””. Wewnątrz napisu można stosować sekwencje specjalne \n, \r, \” , \\ oznaczające kolejno: znaki nowej linii, znak powrotu karetki, znak ”, znak \. Można też wstawiać dowolne znaki w kodzie Unicode poprzez sekwencje *u<4-znakowa liczba szesnastkowa>*.

Przykład:

”Ko\u0144” = ”Koń”

2.1.3 Stałe

Stałe symboliczne, które reprezentują z góry określoną wartość. Są to:

- false, wartość logiczna false(fałsz)
- true, wartość logiczna true(prawda)

- null, wartość null(pusta)
- stałe $nazwa, gdzie *nazwa* określa wartość stałej. Są to wartości używane w wywołaniach funkcji i akcji. Zostały opisane w rozdziałach dotyczących konkretnych akcji i funkcji.

2.1.4 Identyfikatory

Identyfikatory są napisami, w których pierwszym znakiem jest litera, a kolejnymi znakami mogą być litery, cyfry i znak _. Znaczenie identyfikatora wynika z kontekstu jego użycia. W sensie wartości i zachowania w trakcie wyliczania wartości wyrażenia są identyczne z tekstami.

Przykłady

Variable (Limit_1) – w wywołaniu funkcji *Variable* identyfikator *Liimit_1* oznacza nazwę zmiennej

Limit_1 + ”_Hi” – operator *+* łączy identyfikator z napisem dając w efekcie napis *Limit_1_Hi*

2.1.5 Wywołania funkcji

Wywołania funkcji i akcji mają postać:

n*azwa(argument_1, …, argument_n)*

Ilość i znaczenie argumentów są zależne od nazwy funkcji. Argumenty wywołania mogą być również wyrażeniami. Funkcje zostały szczegółowo opisane w osobnych rozdziałach.

2.2 Operatory

Operatory określają operacje, które należy wykonać na elementach wyrażeń. Kolejność wykonania operacji zależy od priorytetów operatorów i kolejności ich użycia (w przypadku operatorów o identycznym priorytecie). Można sterować kolejnością użycia operatorów przy pomocy nawiasów okrągłych (i). Poniższe rozdziały opisują operatory działań w kolejności malejących priorytetów:

2.2.1 ~ ! - (unarny)

Operatory negacji bitowej, negacji logicznej i zmiany znaków.

Przykłady:

~0x000f = 0xfff0

!(1==0) = false

2.2.2 * / %

Operatory mnożenia, dzielenia i dzielenia modulo. Operacja dzielenia na liczbach całkowitych zwraca w wyniku liczbę całkowitą.

Przykłady

17/5 = 3

17%5 = 2

2.2.3 + -

Operatory dodawania i odejmowania. Operator dodawania w przypadku argumentów tekstowych wykonuje operacje łączenia tekstów.

Przykłady:

5 + 9 = 14

”5” + 9 = ”59”

2.2.4 << >>

Operatory przesunięć bitowych w lewo i prawo.

Przykłady:

0x10 <<2 = 0x1000

2.2.5 < <= > >=

Operatory porównania mniejsze, mniejsze lub równe, większe, większe lub równe.

2.2.6 == !=

Operatory porównania równe, różne. Operatory mogą być używane do porównywania tekstów.

2.2.7 &

Operator iloczynu bitowego.

Przykłady:

0xf0 & 0x11 = 0x10

2.2.8 ^

Operator alternatywy wykluczającej (xor).

Przykłady:

0xf0 ^ 0x11 = 0xe1

2.2.9 |

Operator sumy bitowej.

Przykłady:

0xf0 & | 0x11 = 0xf1

2.2.10 &&

Operator iloczynu logicznego.

Przykłady:

1==0 && 2==2 = false

2.2.11 ||

Operator sumy logicznej.

Przykłady:

1==0 || 2==2 = true

2.2.12 ?:

Operator wyrażenia logicznego postaci *warunek*?*wyrażenie_gdy_prawda*:*wyrażenia_gdy_fałsz*. W zależności od wyniku sprawdzenia warunku wyliczane jest jedno z dwóch podwyrażeń.

Przykład

1==0 ? ”tak” : ”nie” = ”nie”

3 Wyliczanie wyrażeń

Asix.Evo stosuje optymalizację wyliczania wartości wyrażeń. Oznacza to, że wyrażenia są przeliczane, tylko wtedy, gdy jest to niezbędne. W przypadku wyrażeń użytych we właściwościach obiektów, obliczenie następuje, gdy zmieni się któryś z elementów wyrażenia.

Attribute(z1, LimitHi)

W tym przypadku obliczenie wyrażenia następuje tylko w momencie otwarcia diagramu lub przy zmianie bazy definicji zmiennych.

10 * Variable (z2)

Przeliczenie następuje tylko w momentach zmiany wartości zmiennej *z2*.

3.1 Typy i konwersja wartości

W trakcie wykonywania operacji następuje automatyczna konwersja wartości argumentów. Sposób konwersji zależy od użytego operatora lub funkcji oraz od typu argumentów.

Zasady konwersji są następujące:

- W przypadku działań arytmetycznych uzgadniane są typy argumentów do typu o większym zakresie wartości.

  12 + 10.10

wynik 22.10, pierwszy argument konwertowany na liczbę zmiennoprzecinkową

- Teksty użyte w wyrażeniach są z reguły konwertowane na liczby, za wyjątkiem operacji wykonywanych bezpośrednio na tekstach.

  10 * Attribute(z1, LimitHi)

atrybut *LimitHi* zmiennej zostanie skonwertowany na tekst i pomnożony przez 10

10 + 1

wynik 11, operacja dodania dwóch liczb

”10” + 1

wynik 101, operacja połączenia tekstów

- W miejscu użycia wartości logicznych, można używać wartości liczbowych. Wartości różne od zera oznaczają *true*, równe zero oznaczają *false*. Można też użyć tekstów *”true”* i *”false”*.

Variable(z1) ? red : blue

Jeżeli wartość zmiennej *z1* jest różna od zera, wynikiem będzie tekst *”red”*

- Identyfikatory są konwertowane na teksty.

Ident1 + 1

wynikiem jest *ident11* ze względu na wykonanie tekstowej operacji +

W niektórych wypadkach niezbędna może okazać się jawna konwersja typu wartości. Służą do tego funkcje rodziny *To….*

ToDouble(Variable(z1))/Variable(z2)

Jeżeli zmienne *z1* i z2 są typu całkowitego, to wynik zwykłego dzielenia byłby też liczba całkowitą. W wyrażeniu użyto jawnej konwersji pierwszej zmiennej, aby uzyskać dokładny wynik dzielenia.

ToUint32(Attribute(z1, LimitHi)) + 10

Jawna konwersja atrybutu na funkcje powoduje, że operator + ma znaczenie dodawania arytmetycznego, a nie łączenia tekstów.

3.2 Kontekst użycia

Istotne znaczenie dla procesu wyliczenia wyrażenia ma również kontekst jego użycia. Wyrażenie użyte w definicji właściwości obiektu ma kontekstowy dostęp do definicji obiektu i diagramu, w którym zostało użyte. Typową informacją kontekstową jest nazwa zmiennej głównej obiektu.

Zestaw danych kontekstowych zależy od miejsca użycia wyrażenia, np. wyrażenia funkcji przeliczających mają jedynie dostęp do zmiennej, w definicji której zostały użyte.

Informacje kontekstowe są dziedziczone. Wyrażenie użyte we właściwości globalnej nie posiada własnego kontekstu. Jednak, użycie tej właściwości globalnej w definicji właściwości obiektu diagramu spowoduje, że wyrażenie otrzyma kontekst obiektu.

Variable()

Powyższe wyrażenie jest legalne we właściwości obiektu i oznacza pobranie wartości zmiennej głównej obiektu. Nie jest jednak legalne w treści akcji operatorskiej użytej w definicji skrótu klawiszowego.

Variable(”#_hi”)

Powyższe wyrażenie też jest legalne, tylko w kontekście (bezpośrednim lub dziedziczonym) obiektu diagramu. Znak specjalny # oznacza nazwę zmiennej głównej, w efekcie funkcja pobiera wartość zmiennej o nazwie utworzonej przez dodanie sufiksu *_hi* do nazwy zmiennej głównej.

3.3 Diagnostyka wyrażeń

W przypadku problemów w prawidłowym skonstruowaniu wyrażenia polecana jest kontrola komunikatów rejestrowanych w panelu *Komunikaty*. W przypadku błędu w trakcie liczenia wyrażenia pojawiają się tam komunikaty opisujące rodzaj błędu i miejsce jego wystąpienia.

Inną użyteczną techniką jest użycie wyrażenia lub jego fragmentu we właściwości *Tekst* obiektu *Tekst* – pozwala to na wizualną kontrolę wyniku wyrażenia.

4 Spis funkcji

Dostęp do elementów aplikacji

Asix6Attribute – zwraca wartość wskazanego atrybutu zmiennej procesowej w postaci liczbowej. Jeżeli atrybut nie jest liczbą, to jest traktowany jako nazwa zmiennej i zwracana jest wartość tej zmiennej.

Attribute – zwraca wartość wskazanego atrybutu zmiennej procesowej

ExistsAttribute – sprawdza w bazie definicji zmiennych aplikacji, czy używany jest podany atrybut

LocalProperty – zwraca wartość właściwości obiektu lub diagramu

LocalPropertyStatus – zwraca status OPC związany z właściwością obiektu lub diagramu

Parameter – zwraca wartość parametru użytego w diagramie lub wzorcu

Parameters – zwraca ciąg nazw i wartości wszystkich parametrów używanych w diagramie, wzorcu lub menu

Property – zwraca wartość właściwości globalnej

Text – zwraca tekst z puli tekstów wielojęzycznych

Dostęp do wartości bieżących zmiennych procesowych

Asix6Limit - sprawdza warunek przekroczenia limitu wartości zmiennej procesowej zgodnie z algorytmem stosowanym w aplikacjach Asix6

CheckStateMask – testuje, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest wartościami liczbowymi maski i spodziewanej wartości.

CheckStateTemplate - testuje, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest w postaci tekstowej.

ExistsVariable – sprawdza, czy zmienna procesowa o podanej nazwie jest zdefiniowana

GetStateDescription – zwraca opis stanu zmiennej procesowej w oparciu o informacje znajdujące się w bazie zmiennych

GetStateText – zwraca nazwę stanu zmiennej procesowej w oparciu o informacje znajdujące się w bazie zmiennych

NewValue - zwraca nową wartość zmiennej procesowej. Funkcja używana tylko w trakcie wykonywania operacji sterującej w funkcji przeliczającej zapisu.

RawValue – zwraca wartość surową zmiennej procesowej, bez wykonania przeliczeń zdefiniowanych w bazie definicji zmiennych

TableElement – zwraca wskazany element z wartości zmiennej procesowej typu tablicowego

TableLength – zwraca liczbę elementów w zmiennej procesowej typu tablicowego

VarChanged – informuje, czy wartość zmiennej uległa zmianie w ostatnim okresie czasu

Variable – zwraca bieżąca wartość zmiennej procesowej po zastosowaniu przeliczeń zdefiniowanych w bazie definicji zmiennych

VarIsGood – informuje, czy status wartości zmiennej procesowej jest dobry.

VarIsNotGood – informuje, czy status wartości zmiennej procesowej nie jest dobry.

VarStatus - zwraca status wartości zmiennej procesowej

VarStringValue – zwraca wartość tekstową zmiennej procesowej, utworzoną na podstawie formatu zdefiniowanego w bazie definicji zmiennych

VarTime – zwraca czas ostatniej aktualizacji wartości zmiennej procesowej

Dostęp do wartości historycznych zmiennych procesowych

HorizonAggregate – zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu liczonego względem chwili bieżącej.

RangeAggregate – zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego zakresu czasu.

Funkcje stanu obiektów i diagramów

HasWaitingControl - informuje, czy znajduje się w stanie oczekiwania na wysłanie sterowania

IsActive - informuje, czy obiekt jest aktywny, czyli może wykonywać operacje sterujące

IsBlinkOff - informuje o aktualnej fazie migotania (zwraca cyklicznie wartości *true* lub *false)*

IsDiagramActive - informuje, czy diagram o podanej nazwie jest aktualnie diagramem aktywnym (bieżącym)

IsDiagramActivePar - informuje, czy diagram o podanej nazwie i zestawie parametrów jest aktualnie diagramem aktywnym (bieżącym)

IsDiagramOpened - informuje, czy diagram o podanej nazwie jest uruchomiony

IsSelected - informuje, czy obiekt jest aktualnie wybrany (jest obiektem bieżącym)

IsWindowOpened - informuje, czy okno o podanej nazwie jest uruchomione

Wsparcie dla systemu alarmów

AlarmsControler – funkcja zwraca nazwę aktywnego kontrolera

AlarmsGroupState – zwraca status grupy alarmów, pozwala sprawdzić, czy w grupie są alarmy aktywne lub/i niepotwierdzone.

AlarmsMode – informuje o aktualnym trybie pracy systemu alarmów

DeadBand – funkcja wykorzystywana w wykrywaniu alarmów, kontroluje przekraczanie limitów z histerezą

DelayedState – funkcja wykorzystywana w wykrywaniu alarmów, wykrywa brak przewidywanych zmian stanu

IsAlarm – informuje, czy alarm znajduje się w logu alarmów aktywnych i czy jest aktywny(niezakończony)

IsAlarmExcluded – informuje, czy wskazany alarm jest aktualnie wykluczony

IsAlarmUnaccepted – informuje, czy alarm znajduje się w logu alarmów aktywnych i czy jest niepotwierdzony

RateOfChange – funkcja wykorzystywana w wykrywaniu alarmów, monitoruje prędkość zmian wartości wyrażenia

Funkcje konwersji współrzędnych

AbsToRelX – konwertuje współrzędną poziomą lub szerokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach

AbsToRelY – konwertuje współrzędną pionową lub wysokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach

RelToAbsX – konwertuje współrzędną poziomą lub szerokość z wartości względnej w pikselach na wartość bezwzględną (0 do 1 000 000)

RelToAbsY – konwertuje współrzędną pionową lub wysokość z wartości względnej w pikselach na wartość bezwzględną (0 do 1 000 000)

XOnScreen - przelicza współrzędną poziomą z wartości względnej wskazanego monitora na wartość bezwzględną dla pełnego pulpitu

YOnScreen - przelicza współrzędną pionową z wartości względnej wskazanego monitora na wartość bezwzględną dla pełnego pulpitu

Kontrola stanu klawiatury i myszki

CursorX – zwraca współrzędną X aktualnego położenia kursora w ramach diagramu

CursorY – zwraca współrzędną Y aktualnego położenia kursora w ramach diagramu

IsControlPressed – informuje, czy klawisz *Control* jest wciśnięty

IsAltPressed – informuje, czy klawisz *Alt* jest wciśnięty

IsShiftPressed – informuje, czy klawisz *Shift* jest wciśnięty

IsMouseOver – informuje, czy kursor myszki znajduje się nad obiektem

IsMousePressed – informuje, czy przyciśnięty jest klawisz myszki i kursor myszki znajduje się nad obiektem

LastKeyPressed – zwraca opis ostatnio przyciśniętego klawisza na klawiaturze

Operacje na łańcuchach tekstowych

Concat – łączy wiele ciągów znaków w pojedynczy napis

Format – zwraca napis utworzony zgodnie z podanym formatem

Replace – zamienia wszystkie wystąpienia wskazanego podciągu na inny ciąg znaków

Substring – zwraca napis będący fragmentem napisu podanego w wywołaniu funkcji

Funkcje arytmetyczne

Cos – wylicza wartość funkcji cosinus

E – zwraca wartość liczby e

Log – wylicza wartość logarytmu naturalnego

Log10 – wylicza wartość logarytmu dziesiętnego

PI – zwraca wartość liczby pi

Pow – wylicza wartość liczby podniesionej do podanej potęgi

Round – zaokrągla liczbę do podanej liczby cyfr ułamkowych

Sin – wylicza wartość funkcji sinus

Sqrt – wylicza wartość pierwiastka kwadratowego

Tan – wylicza wartość funkcji tangens

Funkcje konwersji wartości

CheckOPCStatus – pozwala na uproszczone testowanie wartości statusu OPC

Color – tworzy wartość koloru na podstawie podanych barw podstawowych RGB

FromAsix6Date – konwertuje liczbową wartość daty i czasu aplikacji Asix6 na format daty i czasu

FromAsix6Time – konwertuje liczbową wartość zakresu czasu aplikacji Asix6 na format zakresu czasu

OPCTime – tworzy wartość daty i czasu po zastosowaniu modyfikacji opisanej rozszerzonym wyrażeniem formatującym OPC

ToAsix6Date – konwertuje datę i czas na liczbową wartość daty i czasu aplikacji Asix6

ToAsix6Time – konwertuje zakres czasu na liczbową wartość zakresu czasu aplikacji Asix6

ToBool – konwersja na wartość logiczną *true/false*

ToDateTime - konwertuje wartość do postaci daty i czasu

ToDouble – konwersja na liczbę zmiennoprzecinkową podwójnej precyzji

ToInt8 – konwersja na 8-bitową liczbę całkowitą ze znakiem

ToInt16 – konwersja na 16-bitową liczbę całkowitą ze znakiem

ToInt32 – konwersja na 32-bitową liczbę całkowitą ze znakiem

ToInt64 – konwersja na 64-bitową liczbę całkowitą ze znakiem

ToSingle – konwersja na liczbę zmiennoprzecinkową pojedynczej precyzji

ToString – konwertuje wartość liczbową interpretowaną jako sekwencję znaków Ascii na tekst

ToTimeSpan – konwertuje wartość do postaci zakresu czasu

ToUInt8 – konwersja na 8-bitową liczbę całkowitą bez znaku

ToUInt16 – konwersja na 16-bitową liczbę całkowitą bez znaku

ToUInt32 – konwersja na 32-bitową liczbę całkowitą bez znaku

ToUInt64 – konwersja na 64-bitową liczbę całkowitą bez znaku

ValueOrDefault – zwraca wartość lub wartość domyślną

XorColor - zwraca dopełnienie (XOR) podanego koloru

Pomocnicze

ApplicationChanged – informuje, czy aplikacja została zaktualizowana

AsserviceRead – odczyt wartości pola licznika programu AsService

ChannelState – zwraca status pracy kanału komunikacyjnego

Evaluate - wyznacza wartość wyrażenia podanego w parametrze funkcji

HasRole – sprawdza w systemie uprawnień aplikacji, czy aktualny użytkownik pełni daną rolę

IsActiveNote – sprawdza, czy z określonym segmentem skojarzone są jakieś aktywne notatki operatora

IsBrowser – sprawdza, czy aplikacja jest uruchomiona w oknie przeglądarki

IsScriptWorking – sprawdza, czy jest uruchomiony skrypt o podanej nazwie

IsWorking – sprawdza, czy jest nawiązane lub można nawiązać połączenie z określoną stacją zdefiniowaną w ustawieniach aplikacji

ToFullPath – zwraca bezwzględną ścieżkę do pliku dla ścieżki podanej względem katalogu aplikacji

5 Opisy funkcji

5.1 AbsToRelX

Przeznaczenie

Funkcja konwertuje współrzędną poziomą lub szerokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach. Wartości bezwzględne są przechowywane w definicjach współrzędnych. Przejście na wartości względne wykonywane jest w trakcie wyświetlania obiektu i jest dostosowywane do aktualnych rozmiarów diagramu. Współrzędna bezwzględna 1 000 000 odpowiada prawej krawędzi diagramu. Funkcja używana jest w przypadku implementacji ruchu lub zmiany rozmiaru obiektów.

Składnia

AbsToRelX(*współrzędna_X*)

Parametry

współrzędna_X

Współrzędna X lub szerokość obiektu w jednostkach bezwzględnych, którą należy przeliczyć na wartość względną.

Zobacz też

Asix.Evo - Techniki budowy diagramów.PDF/CHM*, 13.Animacja ruchu i zmiana rozmiaru obiektów*

funkcje: *AbsToRelY*, *RelToAbsX*, *RelToAbsY*

5.2 AbsToRelY

Przeznaczenie

Funkcja konwertuje współrzędną pionową lub wysokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach. Ogólne zasady użycia są takie same jak w przypadku funkcji *AbsToRelX*.

Składnia

AbsToRelY(*współrzędna_Y*)

Parametry

współrzędna_Y

Współrzędna Y lub wysokość obiektu w jednostkach bezwzględnych, którą należy przeliczyć na wartość względną.

Zobacz też

Asix.Evo - Techniki budowy diagramów.PDF/CHM*, 13.Animacja ruchu i zmiana rozmiaru obiektów*

funkcje: *AbsToRelX*, *RelToAbsX*, *RelToAbsY*

5.3 AlarmsControler

Przeznaczenie

Funkcja zwraca nazwę aktywnego kontrolera w wybranej domenie alarmów. W przypadku terminali podpiętych za pośrednictwem pomostu zwracana jest nazwa użytego pomostu.

Składnia

AlarmsControler ()

AlarmsControler (*nazwa_domeny*)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny, używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *AlarmsMode*

5.4 AlarmsCount

Przeznaczenie

Funkcja zwraca ilość alarmów w wybranej grupie alarmów lub w całej domenie. W zależności od sposobu wywołania, można uzyskać ilość alarmów aktywnych, niepotwierdzonych lub aktywnych niepotwierdzonych.

Składnia

AlarmsCount (*nazwa_domeny*, *selektor_grupy*, *rodzaj_alarmów*)

AlarmsCount (*selektor_grupy*, *rodzaj_alarmów*)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

selektor_grupy

Parametr określa podzbiór alarmów, którego stan ma zostać zwrócony. Definicja parametru ma postać tekstu złożonego z par <identyfikator_atrybutu_grupującego>=<wartość_atrybutu> rozdzielonych znakami średnika. Użycie pustego parametru wywołania oznacza zwrócenie ilości alarmów dla całej domeny.

rodzaj_alarmów

Parametr określa typ alarmów, które mają zostać policzone. Parametr musi być jedną z poniższych stałych:

$Active – liczone są alarmy aktywne (niezakończone), niezależnie od ich stanu potwierdzenia.

$ActiveUnaccepted – liczone są alarmy aktywne (niezakończone), które nie zostały jeszcze potwierdzone przez operatora.

$Unaccepted – liczone są alarmy, które nie zostały jeszcze potwierdzone przez operatora.

Zobacz też

Funkcja AlarmsGroupState

5.5 AlarmsGroupState

Przeznaczenie

Funkcja zwraca zbiorczy stan wybranego zbioru alarmów lub całej domeny. Funkcja pozwala określić, czy w kontrolowanym zbiorze alarmów znajdują się alarmy aktywne (niezakończone) oraz alarmy niepotwierdzone. Interpretację zwracanej wartości opisuje poniższa tabelka.

Bit 2 **Istnieje alarm aktywny niepotwierdzony**	**Bit 1** **Istnieje alarm niezakończony**	**Bit 0** **Istnieje alarm niepotwierdzony**	**Wartość liczbowa**	**Interpretacja**
0	0	0	0	Brak alarmów w logu alarmów aktywnych
0	0	1	1	Istnieją tylko alarmy zakończone niepotwierdzone
0	1	0	2	Istnieją tylko alarmy aktywne potwierdzone
0	1	1	3	Mieszanka alarmów zakończonych niepotwierdzonych i aktywnych potwierdzonych
1	1	1	7	Mieszanka alarmów zakończonych niepotwierdzonych, aktywnych potwierdzonych i aktywnych niepotwierdzonych

W zależność od potrzeb zwracany wynik można testować pod kątem ustawionych bitów lub wartości liczbowych.

Należy pamiętać, że wartość funkcji wyliczana jest na podstawie logu alarmów aktywnych. Oznacza to, że obsługa alarmów zakończonych jest zależna od konfiguracji logu. Jeżeli alarmy zakończone są natychmiast usuwane z logu alarmów aktywnych, to funkcja nigdy nie zwróci stanu oznaczającego, że są niepotwierdzone alarmy zakończone. Analogicznie, jeżeli alarmy zakończone są usuwane po zadanym czasie, to informacja o ich niepotwierdzeniu zostanie utracona.

Składnia

AlarmsGroupState (*selektor_grupy*)

AlarmsGroupState (*nazwa_domeny*, *selektor_grupy*)

Parametry

selektor_grupy

Parametr określa podzbiór alarmów, którego stan ma zostać zwrócony. Definicja parametru ma postać tekstu złożonego z par <*identyfikator_atrybutu_grupującego*>=<*wartość_atrybutu*> rozdzielonych znakami średnika. Użycie pustego parametru wywołania oznacza zwrócenie stanu całej domeny alarmów.

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Przykład

=AlarmsGroupState("Strefa=B1;Strefa=B2;Typ=Elektryczna")&4

Funkcja *AlarmsGroupState* zwraca zbiorczy stan dla alarmów, których atrybut grupujący *Strefa* jest równy *B1* lub *B2* i atrybut *Typ* jest równy *Elektryczna.* Całe wyrażenie natomiast zwraca wartość *true*, gdy w zbiorze kontrolowanych alarmów jest przynajmniej jeden alarm aktywny i niepotwierdzony.

Zobacz też

Funkcja *IsAarm*, *IsAlarmUnaccepted*, *AlarmsCount*

5.6 AlarmsMode

Przeznaczenie

Funkcja zwraca tryb pracy stanowiska w ramach wybranej domeny alarmów. Funkcja może zwrócić poniższe wartości:

1. – stanowisko nie bierze udziału w pracy domeny
2. – stanowisko jest kontrolerem aktywnym
3. – stanowisko jest kontrolerem pasywnym

4. – stanowisko jest terminalem lub pomostem

5. – stan stanowiska jest nieustalony, występuje w trakcie startu domeny oraz poszukiwania kontrolera aktywnego

Składnia

AlarmsMode ()

AlarmsMode (nazwa_domeny)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *AlarmsControler*

5.7 ApplicationChanged

Przeznaczenie

Funkcja zwraca wartość logiczną, informującą czy aplikacja została zaktualizowana poprzez mechanizm okresowej synchronizacji definicji aplikacji.

Składnia

ApplicationChanged()

5.8 Asix6Attribute

Przeznaczenie

Funkcja pobiera z bazy definicji zmiennych wartość tekstową wskazanego atrybutu zmiennej procesowej i konwertuje ją na wartość liczbową. Jeżeli atrybut nie jest liczbą, to jest traktowany jako nazwa zmiennej procesowej i zwracana jest wartość tej zmiennej. Funkcja

typowo stosowana w warunkach stanów obiektów, szczególnie w trakcie konwersji aplikacji Asix6.

Składnia

Asix6Attribute (*nazwa_atrybutu*)

Asix6Attribute (*nazwa_zmiennej*, *nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych i przetworzyć zgodnie z opisanym wyżej algorytmem.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *Attribute*, *Asix6Limit*

5.9 Asix6Limit

Przeznaczenie

Funkcja zwraca wartość logiczną, której wartość jest rezultatem sprawdzenia warunku przekroczenia limitu wartości zmiennej procesowej zgodnie z algorytmem stosowanym w aplikacjach Asix6. Funkcja jest typowo stosowana w warunkach stanów obiektów. Szczegółowy algorytm wyliczania wyniku funkcji jest następujący:

- Jeżeli wartość atrybutu jest liczbą, to wartość zmiennej procesowej jest porównywana bezpośrednio z wartością limitu. Funkcja zwraca *true*, gdy wartość zmiennej jest większa od limitu (dla atrybutów *LimitHi* i *LimitHiHi)* lub wartość zmiennej jest mniejsza od limitu (dla atrybutów *LimitLo* i *LimitLoLo).*
- Jeżeli wartość atrybutu jest nazwą zmiennej, to jako próg limitu pobierana jest wartość tej zmiennej. Dalsze działanie jest zgodne z punktem poprzednim.
- Jeżeli wartość atrybutu ma postać *nazwa_zmiennej|maska_bitowa*, to pobierana jest wartość zmiennej podanej w atrybucie i sprawdzane jest, czy

w tej wartości ustawiony jest przynajmniej jeden z jedynkowych bitów maski bitowej.

- Jeżeli wartość atrybutu ma postać *nazwa_zmiennej&maska_bitowa*, to pobierana jest wartość zmiennej podanej w atrybucie i sprawdzane jest, czy w tej wartości ustawione są wszystkie jedynkowe bity maski bitowej.

Składnia

Asix6Limit (nazwa_atrybutu)

Asix6Limit (nazwa_zmiennej, nazwa_atrybutu)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych i przetworzyć zgodnie z opisanym wyżej algorytmem. Dopuszczalne nazwy atrybutów to: *LimitLo, LimitLoLo, LimitHi* i *LimitHiHi*.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *Asix6Attribute*

5.10 AsserviceRead

Przeznaczenie

Funkcja służy do odczytu wybranego pola wartości licznika programu AsService.

Składnia

AsserviceRead (*identyfikator_licznika*, *nazwa_pola*)

Parametry

Identyfikator_licznika

Identyfikator licznika program AsService, którego dotyczy wywołanie funkcji. Licznik o podanym identyfikatorze musi być zdefiniowany w bazie danych programu AsService.

nazwa_pola

Nazwa pola licznika, którego wartość ma zostać zwrócona przez funkcję. Dozwolone są poniższe nazwy pól:

Pola związane z definicją i aktualnym stanem licznika	
Name	Tekstowy opis licznika
CounterType	Typ licznika
CounterTypeDesc	Tekstowy opis typu licznika
VarName	Nazwa zmiennej procesowej na podstawie której przeliczana jest wartość licznika
AggrName	Nazwa agregatu używanego do przeliczeń
MainCounter	Flaga True/False określająca, czy licznik jest licznikiem głównym urządzenia
RefreshFreq	Okres przeliczania wartości licznika
StartValue	Początkowa wartość licznika
Value	Aktualna wartość licznika
StartTime	Czas początku zliczania
NoValueTime	Okres czasu braku odczytu poprawnych wartości zmiennej źródłowej
LastReadTime	Czas ostatniego przeliczenia licznika
NextReadTime	Czas planowanego następnego przeliczenia licznika
OperatingLevel	Wartość limitu eksploatacyjnego
CyclicOperatingLevel	Flaga True/False określająca czy wartość limitu eksploatacyjnego jest odtwarzana po każdym resecie
WarningLevel	Wartość limitu ostrzegawczego
WarningLevel2	Wartość drugiego limitu ostrzegawczego
AutoResetReason	Przyczyna automatycznego resetu licznika
AutoResetReasonDesc	Tekstowy opis przyczyny automatycznego resetu licznika

OperatingTask	Czynność eksploatacyjna związana z przekroczeniem limitu licznika
OperatingTaskDesc	Tekstowy opis czynności eksploatacyjnej
Pola związane z definicją urządzenia, do którego licznik należy	
DeviceId	Identyfikator urządzenia
DeviceName	Nazwa urządzenia
DeviceDesc	Tekstowy opis urządzenia
DeviceSymbol	Alfanumeryczny symbol urządzenia
DeviceRegistryNum	Numer ewidencyjny urządzenia
DeviceFactoryNum	Numer fabryczny urządzenia
DeviceProdYear	Rok produkcji urządzenia
DevicePurchaseDate	Data zakupu urządzenia
DevicePowerInst	
DeviceDimensions	Rozmiary urządzenia
DeviceMass	Masa całkowita urządzenia
DeviceVirtual	Flaga True/False określająca czy urządzenie jest wirtualne
DeviceMachine	Nazwa maszyny, do której urządzenie należy
DeviceMachineDesc	Opis tekstowy maszyny
DeviceDepartment	Nazwa wydziału, do którego urządzenie należy
DeviceDepartmentDesc	Opis tekstowy wydziału
DeviceProducer	Producent urządzenia
DeviceProducerDesc	Opis tekstowy producenta
DeviceSupplier	Dostawca urządzenia
DeviceSupplierDesc	Opis tekstowy dostawcy
DeviceLine	Nazwa linii, do której urządzenie należy
DeviceLineDesc	Opis tekstowy linii

DeviceGroup	Nazwa grupy, do której urządzenie należy
DeviceGroupDesc	Opis tekstowy grupy
DevicePowerInst	Moc zasilania urządzania
DevicePowerType	Typ zasilania urządzenia
DevicePowerTypeDesc	Tekstowy opis typu zasilania
DeviceStatus	Status urządzenia
DeviceStatusDesc	Tekstowy opis statusu
DeviceTechCond	Stan techniczny urządzenia
DeviceTechCondDesc	Tekstowy opis stanu technicznego
Pola opisujące ostatni reset licznika	
ArcEndValue	Wartość licznika w momencie resetu
ArcStartValue	Początkowa wartość licznika
ArcStartTime	Moment uruchomienia licznika
ArcEndTime	Moment resetu licznika
ArcNoValueTime	Okres czasu, przez który brakowało odczytu poprawnych danych źródłowych
ArcProtocolNo	Numer protokołu związanego z resetem licznika
ArcExecutedBy	Osoba wykonująca reset
ArcResetReason	Przyczyna resetu licznika
ArcResetReasonDesc	Opis tekstowy przyczyny resetu

5.11 Attribute

Przeznaczenie

Funkcja pobiera z bazy definicji zmiennej wartość wskazanego atrybutu zmiennej procesowej.

Składnia

Attribute (*nazwa_atrybutu*)

Attribute (*nazwa_zmiennej*, *nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Przykład

Attribute(LimitLoLo)

Pobierany jest atrybut *LimitLoLo* kontekstowej zmiennej procesowej. Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

@LimitLoLo

Zobacz też

Funkcja *ExistsAttribute*

5.12 ChannelState

Przeznaczenie

Funkcja zwraca status pracy kanału komunikacyjnego. Możliwe są poniższe wartości statusu:

0 – kanał pracuje normalnie

1 – kanał w fazie inicjalizacji (nie pracuje)

2 – błąd komunikacji w kanale fizycznym (kanale Asmena) obsługiwanym za pośrednictwem drajwera Asix6

3 – praca w kanale redundantnym kanału Asmena obsługiwanego za pośrednictwem drajwera Asix6

4 – sterownik PLC, z którym komunikuje się drajwer kanału fizycznego (kanału Asmena) obsługiwanego za pośrednictwem drajwera Asix6, jest w stanie STOP

5 - kanał fizyczny (kanał Asmena) obsługiwany za pośrednictwem drajwera Asix6 jest wyłączony z działania

6 – kanał pracuje na drajwerze redundantnym

30 – inny błąd kanału

31 – niepoprawna nazwa kanału w wywołaniu funkcji

32 – stan kanału nie został jeszcze określony

W przypadku użycia kanału Asmena w ramach serwera danych (drajwer Asix6), do określenia jego stanu można również wykorzystać funkcję przeliczającą Asmena STATUS_KANALU.

Składnia

ChannelState(*nazwa_kanału*)

Parametry

nazwa_kanału

Nazwa kanału komunikacyjnego, którego status należy zwrócić.

5.13 CheckOPCStatus

Przeznaczenie

Funkcja służy do testowania wartości statusu OPC. Funkcja zwraca wartość logiczną informującą o tym, czy status spełnia podany warunek. Rodzaj testu jest definiowany przez podanie w parametrze wywołania jednej ze stałych rodziny *$Opc….*

Alternatywą dla funkcji *CheckOPCStatus* są bezpośrednie testy bitowe na wartości statusu. Dla prostych testów poprawności wartości zmiennej procesowej można stosować funkcje *VarIsGood* i *VarIsNotGood*.

Składnia

CheckOPCStatus(*wartość_statusu, rodzaj_testu*)

Parametry

wartość_statusu

Status Opc, którego wartość należy sprawdzić. Typowo status Opc pozyskiwany jest za pomocą funkcji *VarStatus* lub *LocalPropertyStatus*.

rodzaj_testu

Jedna ze stałych rodziny *$Opc…*, która określa rodzaj wykonywanego testu wartości statusu. Pełną listę stałych i ich znaczenie można znaleźć w edytorze wyrażeń w menu *Stałe*.

Przykład

CheckOpcStatus(LocalPropertyStatus(Text), $OpcQualityGood)

Funkcja sprawdza czy wartość wyliczona w właściwości *Text* obiektu jest poprawna.

Zobacz też

Funkcje *VarStatus VarIsGood VarIsNotGood LocalPropertyStatus*

5.14 CheckStateMask

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o tym, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest wartościami liczbowymi maski i spodziewanej wartości. W pierwszym kroku na wartości testowanej zmiennej wykonywana jest operacja iloczynu bitowego z wartością maski. Jeżeli tak uzyskana wartość jest równa wartości spodziewanej, to zwracana jest wartość *true*.

Funkcja jest typowo stosowana w warunkach stanów obiektów.

Składnia

CheckStateMask(*maska_wartości, wartość_stanu*)

CheckStateMask (*nazwa_zmiennej, maska_wartości, wartość_stanu*)

Parametry

maska_wartości

Maska bitowa, wykonywana jest operacja bitowego iloczynu z wartością zmiennej monitorowanej

wartość_stanu

Spodziewana wartość po zastosowaniu maski

nazwa_zmiennej

Nazwa zmiennej, której wartość jest sprawdzana. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, używana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

CheckStateMask(0xf,3)

Funkcja zwraca *true,* gdy stan 4 najmłodszych bitów wartości zmiennej kontekstowej jest równy 0011. Pozostałe bity nie mają znaczenia.

CheckStateMask(z1, 4,0)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 0.

CheckStateMask(z1, 4,4)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 1. W tym przypadku identycznie zadziała prostsze wyrażenie:

Variable(z1)&4

Zobacz też

Funkcja *CheckStateTemplate*

5.15 CheckStateTemplate

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o tym, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest w postaci tekstowej. Opis stanu składa się z sekwencji znaków: **0** , **1** **i** **-** , odpowiadających spodziewanym wartościom bitów zmiennej testowanej. Znak **-** oznacza, że wartość odpowiadającego mu bitu nie ma znaczenia.

Składnia

CheckStateTemplate(*wzorzec_stanu*)

CheckStateMask (*nazwa_zmiennej, wzorzec_stanu*)

Parametry

wzorzec_stanu

Tekstowy opis spodziewanej wartości zmienne testowanej.

nazwa_zmiennej

Nazwa zmiennej, której wartość jest sprawdzana. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej używana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

CheckStateTemplate(”0110----”)

Funkcja zwraca *true,* gdy stan bitów od 7 do 4 wartości zmiennej kontekstowej jest równy 0110. Pozostałe bity nie mają znaczenia.

CheckStateTemplate(z1,”1--”)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 1. W tym przypadku identycznie zadziała prostsze wyrażenie:

Variable(z1)&4

Zobacz też

Funkcja *CheckStateMask*

5.16 Color

Przeznaczenie

Funkcja tworzy wartość koloru na podstawie podanych barw podstawowych RGB. Jest to alternatywa do bezpośredniego podania koloru przez jego nazwę.

Składnia

Color (r, g ,b)

Parametry

r

Czerwona składowa koloru, liczba z zakresu 0 – 255

g

Zielona składowa koloru, liczba z zakresu 0 - 255

b

Niebieska składowa kolor, liczba z zakresu 0 - 255

Przykład

Color(Variable(zR), Variable(zG), Variable(zB))

Funkcja wylicza kolor na podstawie wartości trzech zmiennych procesowych.

Zobacz też

Funkcja *XorColor*

5.17 Concat

Przeznaczenie

Funkcja pozwala na utworzenie napisu poprzez złączenie dowolnej liczby napisów składowych. Podobną funkcjonalność można uzyskać przy pomocy operatora +, ale w przypadku wielokrotnego łączenia funkcja C*oncat* jest bardziej efektywna. Inną alternatywą jest użycie funkcji *Format*.

Składnia

Concat (*napis_1, … , napis_n*)

Parametry

napis_1, …, napis_n

Kolejne napisy składowe, które zostaną złączone w napis wynikowy.

Przykład

Concat(VarTime() ,” ”, Variable(), ” ”, (VarIsGood()?OK,Błąd))

Funkcja łączy w pojedynczy napis stempel czasu, wartość i opis statusu kontekstowej zmiennej procesowej. Alternatywne implementacja pokazano poniżej:

VarTime() + ” ” + Variable() + ” ” + (VarIsGood()?OK,Błąd)

Format(“{0} {1} {2}”, VarTime(), Variable(), (VarIsGood()?OK,Błąd))

Zobacz też

Funkcja *Format*

5.18 Cos

Przeznaczenie

Funkcja wylicza wartość funkcji cosinus.

Składnia

Cos (*kąt*)

Parametry

kąt

Argument funkcji cosinus wyrażony w radianach

Zobacz też

Funkcje *Sin*, *Tan*, *Pi*

5.19 CursorX

Przeznaczenie

Funkcja zwraca aktualną współrzędna X położenia kursora myszki w ramach diagramu. Zwracana jest w wartość bezwzględna z zakresu od 0 do 1000000.

Składnia

CursorX ()

Zobacz też

Funkcja *CursorY*

5.20 CursorY

Przeznaczenie

Funkcja zwraca aktualną współrzędna Y położenia kursora myszki w ramach diagramu. Zwracana jest w wartość bezwzględna z zakresu od 0 do 1000000.

Składnia

CursorY ()

Zobacz też

Funkcja *CursorX*

5.21 DeadBand

Przeznaczenie

Funkcja zwraca wartość logiczną, która informuje o przekroczeniu limitu. Sprawdzanie wykonywane jest z histerezą, próg zmiany statusu zależy od kierunku zmiany wartości. Typowo funkcja jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

DeadBand (*wartość_monitorowana, limit_włączenia, limit_wyłączenia*)

Parametry

wartość_ monitorowana

Wyrażenie, którego prędkość jest testowana na przekroczenie limitów

limit_włączenia

Limit wartości monitorowanej, powyżej którego następuje zmiana wartości funkcji z *false* na *true*

limit_wyłączenia

Limit wartości monitorowanej, poniżej którego następuje zmiana wartości funkcji z *true* na *false.*

Przykład

DeadBand (Variable(s1), Variable(low), Variable(high))

Wartość zmiennej *s1* jest sprawdzana pod kątem przekroczenia limitów z uwzględnieniem histerezy. Wartości progowe są dynamicznie zmienne, zależne od aktualnych wartości zmiennych *low* i *high*.

Zobacz też

- Asix.Evo_System_alarmów.PDF/CHM, *5.2. Strategia warunkowa*

- funkcje *RateOfChange*, *DelayedState*

5.22 DelayedState

Przeznaczenie

Funkcja zwraca wartość logiczną, która informuje o tym, że w systemie nie zaszły przewidywane zmiany stanu. Funkcja monitoruje stan warunku początkowego. Jeżeli on zajdzie, to sprawdzane jest, czy w zadanym czasie zostanie spełniony warunek końcowy. Jeżeli nie, to wartość funkcji jest równa *true*. Typowo funkcja jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

DelayedState (*warunek_startu, limit_czasu, warunek_zakończenia*)

Parametry

warunek_ startu

Warunek, którego zmiana na *true* powoduje początek cyklu sprawdzania warunku końcowego.

limit_czasu

Okres czasu w sekundach, w ciągu którego spodziewanie jest spełnienie warunku końcowego.

warunek_zakończenia

Warunek logiczny, który powinien zostać spełniony w zadanym czasie.

Przykład

DelayedState (Variable(st1)==1, 5, Variable(stop)==1)

Od momentu zmiany wartości zmiennej *st1* na wartość 1 przez 5 sekund sprawdzane jest, czy wartość zmiennej *stop* też zmieniła się na 1. Jeżeli nie, to funkcja przyjmie wartość *true.*

Zobacz też

- Asix.Evo_System_alarmów.PDF/CHM, *5.2. Strategia warunkowa*

- *f*unkcje *RateOfChange*, *DeadBand*

5.23 E

Przeznaczenie

Zwraca wartość liczby e.

Składnia

E ()

Zobacz też

Funkcja *Log*

5.24 Evaluate

Przeznaczenie

Funkcja wylicza wartość wyrażenia (ewaluatora) podanego w parametrze funkcji. Parametr jest tekstem, w którym należy stosować składnię definicji właściwości. Funkcja jest używana

wtedy, gdy nie jest z góry znana treść wyliczanego wyrażenia, np. wyrażenie jest odczytywane z atrybutów zmiennej procesowej.

Składnia

Evaluate (wyrażenie)

Parametry

wyrażenie

Treść obliczanego wyrażenia

Przykład

Evaluate(Attribute(a1))

Z atrybutu *a1* zmiennej kontekstowej pobierana jest treść ewaluatora, który zostanie następnie wyliczony. Poniższe przykłady pokazują efekt obliczenia w zależności od definicji atrybutu:

#_s1	wartość zmiennej o nazwie zmiennej kontekstowej z sufiksem *s1*
@Name	nazwa zmiennej głównej
!g1	wartość właściwości globalnej *g1*
=7+4	11
7+4	tekst *7+4*

5.25 ExistsAttribute

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy w bazie definicji zmiennych aplikacji używany jest atrybut o podanej nazwie.

Składnia

ExistsAttribute (*nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, którego obecność w bazie definicji zmiennych jest sprawdzana.

Zobacz też

Funkcja *Attribute*

5.26 ExistsVariable

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy w bazie definicji zmiennych aplikacji zdefiniowana jest zmienna o podanej nazwie.

Składnia

ExistsVariable (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której obecność w bazie definicji zmiennych jest sprawdzana. Można stosować notację sufiksową nazw zmiennych z użyciem przedrostka #.

5.27 Format

Przeznaczenie

Funkcja tworzy napis, który powstaje na podstawie napisu formatującego i zestawu argumentów. Argumenty te są konwertowane na tekst zgodnie z instrukcjami zawartymi w napisie formatującym i wstawiane we wskazane miejsca napisu docelowego. Działanie funkcji jest identyczne z działaniem funkcji *String.Format* platformy .NET. Na witrynie *msdn.microsoft.com* można znaleźć szczegółowy opis działania funkcji *Format* łącznie z opisem struktury napisu formatującego. W przykładach użycia pokazano typowe przypadki działania funkcji.

Składnia

Format(*napis_formatujący, argument_0, argument_1, …*)

Parametry

napis_formatujący

Napis formatujący zgodny z formatem platformy .NET. Fragmenty napisu postaci {x:yyy} oznaczają miejsca wstawienia pozostałych argumentów funkcji, gdzie *x* oznacza numer kolejny argumentu (pierwszy argument ma numer 0), a *yyy* określa szczegóły formatowania obiektu.

argument_1, argument_2, …

Kolejne argumenty wstawiane do napisu wynikowego zgodnie z zasadami podanymi w napisie formatującym.

Przykład

Format (”{0} {1}”,Variable(),Attribute(Unit))

Tworzy połączenie wartości zmiennej, skonwertowanej na tekst w sposób domyślny wynikający z typu wartości, z opisem jednostki pomiaru pobranym z bazy definicji zmiennych, np. „122 kg”

Format("{0:f4}", Variable())

Konwertuje liczbę zmiennoprzecinkową na tekst z zaokrągleniem do 4 cyfr ułamkowych.

Format("{0:x}", Variable())

Konwertuje liczbę na postać tekstową szesnastkową.

Format("{0:hh:mm:ss}", Variable(DateTime))

Tworzy napis zawierający aktualną godzinę, np. 11:30:00

Format("{0:# ###}", Variable())

Formatuje duże liczby wstawiając znak spacji przed cyfrą setek.

Zobacz też

Funkcje *Concat*, *VarStringValue*

5.28 FromAsix6Date

Przeznaczenie

Funkcja przelicza datę i czas wyrażoną liczbą sekund, która upłynęła od 1.1.1970 na wartość typu data i czasu (typ DateTime platformy .NET). Funkcja używana głównie przy konwersji aplikacji z Asix6.

Składnia

FromAsix6Date (*sekundy*)

Parametry

sekundy

Liczba sekund od 1.1.1970.

Zobacz też

Funkcje *ToDateTime*, *ToAsix6Date*, *OPCTime*

5.29 FromAsix6Time

Przeznaczenie

Funkcja przelicza czas wyrażony liczbą milisekund na wartość typu zakres czasu (typ TimeSpan platformy .NET).

Składnia

FromAsix6Time (*milisekundy*)

Parametry

milisekundy

Liczba milisekund.

Zobacz też

Funkcje *ToTimeSpan*, *ToAsix6Time*, *OPCTime*

5.30 GetStateDescription

Przeznaczenie

Funkcja zwraca opis stanu wyrażenia (zmiennej procesowej) w oparciu o informacje znajdujące się w bazie definicji zmiennych. Algorytm działania funkcji jest następujący:

- Określana jest nazwa grupy opisującej stany poprzez odczyt wartości atrybutu *Nazwy stanów* dla zmiennej podanej w wywołaniu funkcji.
- W bazie definicji zmiennych wyszukiwane są wszystkie zmienne, których atrybut *Zestaw stanów*, jest identyczny z nazwą grupy określoną w poprzednim punkcie. Typowo są to zmienne nieaktywne służące tylko do definiowania nazw stanów.
- Wśród zmiennych wybranych w poprzednim punkcie wyszukiwana jest ta, której wartość atrybutu *Wartość_stanu* jest identyczna z wartością stanu przekazanego do wywołania funkcji *GetStateDescription*.

- Wynikiem funkcji jest wartość atrybutu *Opis* zmiennej wybranej w poprzednim punkcie.

Efekt podobny do działania funkcji *GetStateDescription* można uzyskać stosując wzorzec bazujący na obiekcie *Tekst*, w którym zostały zdefiniowane odpowiednie stany.

Składnia

GetStateDescription (*nazwa_zmiennej, wartość_stanu*)

Parametry

nazwa_zmiennej

Nazwa zmiennej służącej do pozyskania opisów stanów z bazy definicji zmiennych . Można używać notacji sufiksowej. Pusta nazwa oznacza użycie zmiennej kontekstowej.

wartość_stanu

Wartość określająca stan, dla którego należy uzyskać opis. W typowym przypadku jest to wyrażenie wykorzystujące wartość zmiennej przekazanej w parametrze *nazwa_zmiennej* (nie jest to jednak obowiązkowe).

Przykład

GetStateDescription (””,Variable())

Funkcja pobiera opisy stanów dla zmiennej kontekstowej, a następnie określa odpowiedni opis na podstawie wartości zmiennej kontekstowej.

Zobacz też

Funkcja *GetStateText*

5.31 GetStateText

Przeznaczenie

Funkcja zwraca tekstowy opis stanu na podstawie dwóch parametrów liczbowych: stanu i podstanu oraz informacji znajdujących się w bazie definicji zmiennych. Typowym zastosowaniem jest tworzenie opisów dla wartości pojedynczych bitów, gdzie stan to numer bitu, a podstan to aktualna wartość bitu. Algorytm działania funkcji jest następujący:

- Pobierana jest wartość atrybutu *Nazwy stanów* dla zmiennej podanej w wywołaniu funkcji.

- Jeżeli pobrany atrybut to ciąg postaci *0=stan1;1=stan2;…* , to zwracana jest nazwa stanu odpowiadająca wartości równej parametrowi podstanu. Ten wariant zbliżony jest działaniu do funkcji *GetStateDescription*, ale nie wymaga dodatkowych zmiennych.
- Jeżeli działanie funkcji nie zostało jeszcze zakończone, to w bazie definicji zmiennych wyszukiwane są wszystkie zmienne, których atrybut *Zestaw stanów*, jest identyczny z wartością atrybutu *Nazwy stanów* pobranego w punkcie pierwszym. Typowo są to zmienne nieaktywne służące tylko do definiowania nazw stanów.
- Wśród zmiennych wybranych w poprzednim punkcie wyszukiwana jest ta, której wartość atrybutu *Wartość_stanu* jest identyczna z wartością stanu przekazanego do wywołania funkcji *GetStateText*.
- Jeżeli atrybut *Nazwy stanów* zmiennej wybranej w poprzednim punkcie stanowi ciąg postaci *0=stan1;1=stan2;…,* to zwracana jest nazwa stanu odpowiadająca wartości równej parametrowi podstanu. W przeciwnym przypadku wartością funkcji jest kompletna wartość atrybutu.

Składnia

GetStateText (*nazwa_zmiennej, wartość_stanu, wartość_podstanu*)

Parametry

nazwa_zmiennej

Nazwa zmiennej służącej do pozyskania opisów stanów z bazy zmiennych . Można używać notacji sufiksowej. Pusta nazwa oznacza użycie zmiennej kontekstowej.

wartość_stanu

Wartość określająca stan, dla którego należy uzyskać opis w przypadku stosowania Wirtualnych zmiennych definiujących opisy.

wartość_podstanu

Parametr określający wartość podstan, stosowana przy wyciąganiu opisu z teksów w postaci *0=stan1;1=stan2;…* W typowym przypadku jest to wyrażenie wykorzystujące wartość zmiennej przekazanej w parametrze *nazwa_zmiennej* (nie jest to jednak obowiązkowe).

Przykład

GetStateText(z1, 3 , Variable(z1)&8)

Zadaniem funkcji jest uzyskanie opis stanu pojedynczego bitu. Przykładowe dane w bazie definicji zmiennych:

- Atrybut *Nazwy stanów* zmiennej *z1* jest równy *OpisyBitów*

- Nieaktywna zmienna *Bit3* ma atrybut *Zestaw stanów* równy *OpisyBitów*, a atrybut *Wartość stanu* równy 3.
- Atrybut *Nazwy stanów* zmiennej *Bit3* ma postać *0=OK;8=Awaria*

W zależności od wartości bitu 3 zmiennej *z1* zostanie zwrócony opis *OK* lub *Awaria*.

Zobacz też

Funkcja *GetStateDescription*

5.32 HasRole

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aktualnie zalogowany użytkownik pełni podaną rolę. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektu, właściwościach *Aktywny* lub *Widoczny* lub operatorskich akcjach warunkowych *Break* i *Perform*.

Składnia

HasRole (*nazwa_roli*)

Parametry

nazwa_roli

Nazwa roli, do której przynależność jest sprawdzana.

Zobacz też

Asix.Evo_Parametryzacja_aplikacji.PDF/CHM, *5. System uprawnień*

5.33 HasWaitingControl

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy obiekt wynikający z kontekstu użycia lub obiekty o zgodnej nazwie znajdują się w stanie oczekiwania na wysłanie sterownia. Funkcja dotyczy obiektów na diagramach synoptycznych, które mogą pracować w trybie opóźnionego

sterowania. Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu obiektu oczekującego na wysłanie sterowania.

Składnia

HasWaitingControl ()

HasWaitingControl (*wzorzec_nazwy*)

Parametry

wzorzec_nazwy

Wzorzec nazwy obiektów, których stan jest kontrolowany. We wzorcu można stosować znaki specjalne * i ?. Parametr „*” sprawdza stan wszystkich obiektów diagramu, nawet tych nieposiadających nazwy. Możliwe jest też użycie jednej z poniższych stałych:

$All – test dotyczy wszystkich obiektów umieszczonych na wszystkich otwartych diagramach

$Window – test dotyczy wszystkich obiektów umieszczonych na diagramach bieżącego okna

$Diagram – test dotyczy wszystkich obiektów umieszczonych na bieżącym diagramie

Użycie bezparametrowego wariantu oznacza test stanu obiektu kontekstowego

Zobacz też

- funkcje *IsActive*, *IsSelected*

- Asix.Evo_Akcje.PDF/CHM, akcje *SendControls, CancelControls*

5.34 HorizonAggregate

Przeznaczenie

Funkcja zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu liczonego względem chwili bieżącej (tzw. agregat kroczący)

Składnia

HorizonAggregate(*nazwa_agregatu, horyzont_czasu, okres_odświeżania*)

HorizonAggregate(*nazwa_zmiennej, nazwa_agregatu, horyzont_czasu, okres_odświeżania*)

Parametry

nazwa_agregatu

Nazwa agregatu, który należy wyliczyć. Dostępne nazwy to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Nazwa *none* oznacza odczyt danych surowych.

horyzont_czasu

Okres wyliczanego agregatu. Liczony zawsze względem chwili bieżącej. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan)*

okres_odświeżania

Okres ponawiania odczytu danych archiwalnych. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan)*

nazwa_zmiennej

Nazwa zmiennej, której wartości historyczne są pobierane. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, używana jest zmienna wynikająca z kontekstu użycia.

Przykład

HorizonAggregate(max,”00:15:00”,”00:00:10)

Funkcja zwraca maksymalną wartość zmiennej kontekstowej z okresu ostatnich 15 minut. Wartość funkcji jest przeliczana co 10 sekund.

Zobacz też

- Funkcje *RangeAggregate*, *ToTimeSpan*

- *Asix.Evo_rodzaje_agregatów_danych_archiwalnych.PDF*

5.35 IsActive

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy obiekt wynikający z kontekstu użycia jest aktywny, czyli może wykonywać operacje sterujące (interaktywne). Wynik funkcji jest bezpośrednio związany z wartością właściwości *Aktywny* obiektu. Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu obiektu.

Składnia

IsActive ()

Zobacz też

Funkcje *IsSelected*, *HasWaitingControl*

5.36 IsActiveNote

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy z określonym segmentem skojarzone są aktualnie jakieś aktywne notatki operatora.

Składnia

IsActiveNote (*identyfikator_segmentu*)

Parametry

identyfikator_segmentu

Identyfikator segmentu, dla którego sprawdzana jest obecność aktywnych notatek. Można stosować znaki specjalne * i ?.

5.37 IsAlarm

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm jest aktualnie aktywny (rozpoczęty i niezakończony).

Składnia

IsAlarm (*identyfikator_alarmu*)

IsAlarm (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status aktywności ma być zwrócony.

nazwa_domeny

Nazwa domeny , do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *IsAlarmUnaccepted*, *AlarmsGroupState*

5.38 IsAlarmExcluded

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm jest aktualnie na liście alarmów wykluczonych z wykrywania.

Składnia

IsAlarmExcluded (*identyfikator_alarmu*)

IsAlarmExcluded (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status wykluczenia ma być zwrócony.

nazwa_domeny

Nazwa domeny, do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Asix.Evo_Akcje.PDF/CHM, akcje *ExcludeAlarm*, *IncludeAlarm*

5.39 IsAlarmUnaccepted

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm występuje w logu alarmów aktywnych i nie został jeszcze potwierdzony przez operatora.

Składnia

IsAlarmUnaccepted (*identyfikator_alarmu*)

IsAlarmUnaccepted (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status potwierdzenia ma być zwrócony.

nazwa_domeny

Nazwa domeny , do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *IsAlarm*, *AlarmsGroupState*

5.40 IsAltPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Alt*. Typowe zastosowania funkcji to uwarunkowanie pewnych działań lub wyglądu diagramu od przyciśnięcia klawisza *Alt*. Funkcja typowa używana w warunkach stanów obiektu, właściwościach *Aktywny* lub *Widoczny* lub operatorskich akcjach warunkowych *Break* i *Perform*.

Składnia

IsAltPressed ()

Przykład

Perform(IsAltPressed(), SetVariable(z1,1), Nothing())

Zmienna *z1* jest sterowana tylko wtedy, gdy w momencie wykonania akcji klawisz *Alt* jest przyciśnięty.

Zobacz też

Funkcje *IsControlPressed*, *IsShiftPressed*

5.41 IsBlinkOff

Przeznaczenie

Funkcja zwraca cyklicznie wartości *true* lub *false* informujące o aktualnej fazie migotania. Funkcja typowa używana w warunkach stanów obiektu lub wyrażeniach warunkowych we właściwościach obiektów. Pozwala to zmieniać wygląd obiektów w zależności od fazy migotania.

Składnia

IsBlinkOff ()

Przykład

IsBlinkOff()?Blue:Red

Wyrażenie zwraca cyklicznie wartości kolorów *Blue* i R*ed*.

Zobacz też

Asix.Evo_Techniki_budowy_diagramów.PDF/CHM, *4.3. Migotanie poprzez zmianę właściwości obiektów*

5.42 IsBrowser

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aplikacja jest uruchomiona w oknie przeglądarki www.

Składnia

IsBrowser ()

5.43 IsControlPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Control*. Zastosowania funkcji są identyczne jak w przypadku funkcji *IsAltPressed*.

Składnia

IsAltPressed ()

Zobacz też

Funkcje *IsAltPressed*, *IsShiftPressed*

5.44 IsDiagramActive

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy diagram o podanej nazwie (lub diagram kontekstowy) jest aktualnie diagramem aktywnym (bieżącym). Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramActive()

IsDiagramActive(*wzorzec_nazwy*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?. Jeżeli nazwa nie jest podana, to funkcja sprawdza aktywność diagramu kontekstowego.

Przykład

IsDiagramActive(”Stac*”)

Funkcja zwraca *true*, jeżeli aktywny jest diagram o nazwie rozpoczynającej się od *Stac*.

Zobacz też

Funkcja *IsDiagramOpened*

5.45 IsDiagramActivePar

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy diagram o podanej nazwie i zestawie parametrów otwarcia jest aktualnie diagramem aktywnym (bieżącym). Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramActivePar(*wzorzec_nazwy*, *parametry*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?.

parametry

Parametry otwarcia diagramu. Podawane w sposób określony w opisie akcji *OpenDiagram*.

Przykład

IsDiagramActivePar(”Hydrant”,”nr=7”)

Funkcja zwraca *true*, jeżeli aktywny jest diagram o nazwie *Hydrant* otwarty z parametrem *nr* równym 7.

Zobacz też

Funkcja *IsDiagramOpened*, *IsDiagramActive*

5.46 IsDiagramOpened

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest otwarty diagram o podanej nazwie i parametrach otwarcia. Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramOpened(*wzorzec_nazwy*)

IsDiagramOpened(*wzorzec_nazwy, parametry_otwarcia*)

IsDiagramOpened(*wzorzec_nazwy, parametry_otwarcia, ekran*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?.

parametry_otwarcia

Parametry otwarcia diagramu, których zgodność jest weryfikowana przez funkcję. Sprawdzana jest zgodność tylko parametrów podanych w wywołaniu. Jeżeli parametry nie są podane, to funkcja sprawdza tylko nazwę diagramu.

ekran

Numer ekranu, na którym sprawdzane jest otwarcie diagramu. Możliwe są poniższe wartości:

-1 - ekran jest dowolny (działanie identyczne jak dla wywołania bez podania numeru ekranu)
0 - ekran na którym znajduje się kursor myszki

większe od 1 - systemowy numer ekranu

Przykład

IsDiagramOpened(”StacX”,”pokoj=11”**)**

Funkcja zwraca *true,* jeżeli otwarty jest diagram o nazwie *StacX* i parametrach otwarcia identycznych z *pokoj=11*.

Zobacz też

Funkcja *IsDiagramActive*

5.47 IsMouseOver

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy kursor myszki znajduje się nad obiektem. Można sprawdzać pozycję nad obiektem kontekstowym lub wskazanym przez nazwę. Funkcja typowo używana w warunkach stanów obiektu w celu zmiany wyglądu obiektu po najechaniu na niego kursorem myszki.

Funkcja wymaga, żeby sprawdzany obiekt miał właściwość *Aktywny* ustawioną na *true*.

Składnia

IsMouseOver (*wzorzec_nazwy*)

IsMouseOver ()

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy obiektów, dla których sprawdzany jest warunek najechania kursorem myszki. Można używać znaków * i ?. Użycie wariantu wywołania bez podania wzorca nazwy oznacza sprawdzanie wyłącznie obiektu kontekstowego.

Zobacz też

- Asix.Evo_Techniki_budowy_diagramów.PDF/CHM, *7.1. Nawigacja poprzez tekstowe łącza*

- Funkcje *IsMousePressed*

5.48 IsMousePressed

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy dowolny klawisz myszki jest wciśnięty, a kursor myszki znajduje się nad obiektem wynikającym z kontekstu użycia.

Funkcja wymaga, żeby sprawdzany obiekt miał właściwość *Aktywny* ustawioną na *true*.

Składnia

IsMousePressed ()

Zobacz też

- - Asix.Evo_Techniki_budowy_diagramów.PDF/CHM, *7.1. Nawigacja poprzez tekstowe łącza*

- Funkcje *IsMouseOver*

5.49 IsSelected

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy obiekt wynikający z kontekstu użycia jest aktualnie wybrany (jest obiektem bieżącym diagramu). Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu wyselekcjonowanego obiektu.

Składnia

IsSelected ()

Zobacz też

Funkcje *IsActive*, *HasWaitingControl*

5.50 IsShiftPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Shift*. Zastosowania funkcji są identyczne jak w przypadku funkcji *IsAltPressed*.

Składnia

IsAltPressed ()

Zobacz też

Funkcje *IsControlPressed*, *IsAltPressed*

5.51 IsScriptWorking

Przeznaczenie

Funkcja służy do sprawdzenia czy jest aktualnie uruchomiony skrypt o podanej nazwie i liście parametrów startowych.

Składnia

IsScriptWorking (*nazwa_skryptu*))

IsScriptWorking (*nazwa_skryptu*, *parametr_1*, … , *parametr_n*)

Parametry

nazwa_skryptu

Nazwa skryptu, którego status pracy jest zwracany.

parametr_1, … , parametr_n

Opcjonalna lista parametrów uruchomienia skryptu.

5.52 IsWindowOpened

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest otwarte okno o podanej nazwie. Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsWindowOpened(*wzorzec_nazwy*)

IsWindowOpened(*wzorzec_nazwy, ekran*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy okna. Można używać znaków * i ?.

ekran

Numer ekranu, na którym sprawdzane jest otwarcie okna. Możliwe są poniższe wartości:

-1 - ekran jest dowolny (działanie identyczne jak dla jednoargumentowej wersji wywołania)
0 - ekran na którym znajduje się kursor myszki

większe od 1 - systemowy numer ekranu

Zobacz też

Funkcja *IsDiagramOpened*

5.53 IsWorking

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest nawiązane lub można nawiązać połączenie z określoną stacją zdefiniowaną w ustawieniach aplikacji. Wartość funkcji jest odświeżana cyklicznie.

Składnia

IsWorking(*nazwa_stanowiska*)

Parametry

nazwa_stanowiska

Nazwa testowanego stanowiska. Nazwa musi być zdefiniowana w ustawieniach aplikacji.

5.54 LastKeyPressed

Przeznaczenie

Funkcja zwraca tekstowy opis ostatnio przyciśniętego klawisza na klawiaturze. Postać opisu jest zgodna z opisami stosowanymi w definicji skrótów klawiszowych. Funkcja jest przeznaczona do użycia w obsłudze zdarzenia *Wciśnięcie klawisza* obiektów. Pozwala uzależnić działanie od rodzaju przyciśniętego klawisza.

Składnia

LastKeyPressed ()

Przykład

Perform (LastKeyPressed()==”Alt+G”, SetVariable (v1, 1), Nothing())

Zmienna v1 jest ustawiana na 1, jeżeli zostały przyciśnięte klawisze Alt i g. Funkcja powinna być użyta w obsłudze zdarzenia *Wciśnięcie klawisza*.

Zobacz też

Standardowe zdarzenia obiektów (Asix.Evo_Elementy_wizualizacji.PDF/CHM)

5.55 Log

Przeznaczenie

Funkcja wylicza wartość logarytmu naturalnego.

Składnia

Log (*argument*)

Parametry

argument

Argument funkcji logarytmu naturalnego

Zobacz też

Funkcje *E*, *Log10*

5.56 LocalProperty

Przeznaczenie

Funkcja zwraca wartość właściwości lokalnej obiektu diagramu. Można pobrać właściwość obiektu kontekstowego lub dowolnego innego wskazanego nazwą.

Składnia

LocalProperty (*nazwa_właściwości*)

LocalProperty (*nazwa_obiektu, nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości lokalnej, którą należy pobrać.

*n**azwa_obiektu***

Nazwa obiektu, którego właściwość lokalną należy pobrać. W przypadku wariantu funkcji bez podania nazwy obiektu, pobierany jest atrybut obiektu wynikającego z kontekstu użycia. Jeżeli funkcja użyta jest w obiekcie znajdującym się w osadzonym wzorcu, to obiekt o danej nazwie najpierw jest szukany w tym samym wzorcu.

Dopiero gdy takiego nie ma, to obiekt szukany jest na diagramie, ale z pominięciem obiektów wchodzących w skład innych wzorców.

Przykład

LocalProperty(o2,"Color")

Pobierana jest definicja właściwości lokalnej *Color* z obiektu o nazwie *o2*.

Zobacz też

Funkcje *Property*

5.57 LocalPropertyStatus

Przeznaczenie

Funkcja zwraca wartość statusu dla właściwości lokalnej obiektu diagramu. Można pobrać status właściwości obiektu kontekstowego lub dowolnego innego wskazanego nazwą.

Z wartością każdej właściwości obiektu związany jest jej status Opc. Status ten w większości przypadków jest stale ustawiony na QualityGood. Jeżeli jednak definicja właściwości odwołuje się bezpośrednio do jakiegoś elementu aplikacji, który status posiada, to ten status jest zwracany. Typowe przypadki to:

a) Odwołania do wartości zmiennych poprzez prefiksy & i # lub samodzielne (nie w złożonych wyrażeniach) wywołania funkcji *Variable* lub *VarStringValue*.

b) Status wyliczenia agregatów historycznych funkcjami *HorizonAggregate* lub *RangeAggregate* (dla samodzielnych wywołań).

c) Odwołania do wartości liczników programu AsService funkcją *AsserviceRead*.

Składnia

LocalPropertyStatus (*nazwa_właściwości*)

LocalPropertyStatus (*nazwa_obiektu, nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości lokalnej, której status należy pobrać.

*n**azwa_obiektu***

Nazwa obiektu, dla którego status właściwości lokalnej należy pobrać. W przypadku wariantu funkcji bez podania nazwy obiektu, pobierany jest status właściwości

obiektu wynikającego z kontekstu użycia. Jeżeli funkcja użyta jest w obiekcie znajdującym się w osadzonym wzorcu, to obiekt o danej nazwie najpierw jest szukany w tym samym wzorcu. Dopiero gdy takiego nie ma, to obiekt szukany jest na diagramie, ale z pominięciem obiektów wchodzących w skład innych wzorców.

Przykład

LocalPropertyStatus(o2,"Text")

Pobierany jest status właściwości lokalnej *Text* z obiektu o nazwie *o2.* W przypadku gdyby właściwość *Text* była obliczana np. wyrażeniem *=HorizonAggregate(max,"00:15:00","00:00:10")* , to pobrany byłby status wyliczenia agregatu historycznego zmiennej procesowej.

Zobacz też

Funkcja *CheckOPCStatus*

5.58 Log10

Przeznaczenie

Funkcja wylicza wartość logarytmu dziesiętnego.

Składnia

Log 10(*argument*)

Parametry

argument

Argument funkcji logarytmu dziesiętnego

Zobacz też

Funkcje *Log*

5.59 NewValue

Przeznaczenie

Funkcja zwraca wartość sterującą zmiennej procesowej, która została przekazana do wysłania w efekcie działań operatora lub skryptów użytkownika. Funkcja używana jest tylko w trakcie wykonywania operacji sterującej w wyrażeniach atrybutu *Funkcja przeliczająca zapisu*. Wynik obliczenia takiego wyrażenia jest wartością przeliczoną wysyłaną do drajwera kanału komunikacyjnego. Funkcja działa na zmiennej kontekstowej, której dotyczy operacja sterująca.

UWAGA: w przypadku korzystania z kanału komunikacyjnego *Asix6,* wartość sterująca jest jeszcze dodatkowo przeliczana przez moduł komunikacyjny Asmen na podstawie atrybutu *Funkcja przeliczająca*.

Składnia

NewValue ()

Zobacz też

Funkcje *RawValue*

5.60 OPCTime

Przeznaczenie

Funkcja tworzy wartość daty i czasu wyliczaną względem chwili aktualnej lub dowolnego innego momentu czasu. Przesunięcie opisane jest rozszerzonym wyrażeniem formatującym OPC.

Składnia

OPCTime (*moment_bazowy, przesunięcie_czasu*)

OPCTime (*przesunięcie_czasu*)

Parametry

moment_bazowy

Chwila czasu, względem którego liczone jest przesunięcie. W przypadku wariantu wywołania bez momentu bazowego przesunięcie jest liczone względem chwili bieżącej.

przesunięcie_czasu

Określa modyfikację momentu bazowego służącą do wyliczenia wyniku funkcji. Przesunięcie opisane jest rozszerzonym wyrażeniem formatującym OPC postaci:

[NOW|SECOND|MINUTE|HOUR|DAY|WEEK|MONTH|YEAR][cykl[/przesunięcie]] [[+|-][liczba][Y|MO|W|D|H|M|S]...]

Pierwsza część wyrażenia określa początkowe przesunięcie, np. NOW oznacza dokładnie moment bazowy, HOUR początek godziny. Klauzule *cykl* i *przesunięcie* pozwalają na implementacje cykli, np. HOUR8/6 pozwala na wyliczenie początku 8-godzinnej zmiany, gdzie pierwsza zmiana zaczyna się o godzinie 8. Końcowa część wyrażenia pozwala dodatkowo przesunąć czas, np. HOUR-15M , oznacza 15 minut przed początkiem godziny.

Przykład

Poniższe przykłady zakładają, że moment bazowy (zadeklarowany jawnie lub domyślnie) to 2011//05/09 11:06:30

NOW (lub pusty)	2011//05/09 11:06:30
NOW-1H	2011//05/09 10:06:30
HOUR	2011//05/09 11:00:00
HOUR8/6	2011//05/09 06:00:00
DAY+12H+15M	2011//05/09 12:15:00
MONTH-1D	2011//04/30 00:00:00

5.61 Parameter

Przeznaczenie

Funkcja zwraca wartość parametru użytego w diagramie lub wzorcu. Działanie zależy od kontekstu użycia. Funkcja użyta wewnątrz wzorca pobiera zawsze wyłącznie parametry wzorca.

Składnia

Parameter (*nazwa_parametru*)

Parametry

nazwa_parametru

Nazwa parametru, który należy pobrać z zestawu parametrów diagramu lub wzorca

Przykład

Parameter(pokoj)

Pobierany jest parametr *pokoj.* Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

%pokoj

5.62 Parameters

Przeznaczenie

Funkcja zwraca ciąg nazw i wartości wszystkich parametrów używanych w bieżącym diagramie, wzorcu lub menu. Format ciągu parametrów jest zgodny z formatem używanym w akcjach *OpenDiagram*, *OpenWindow* i *ShowMenu*. Funkcja ma za zadanie ułatwić przekazywanie parametrów pomiędzy sekwencyjnie otwieranymi diagramami i menu.

Składnia

Parameters ()

Przykład

ShowMenu((menu1,Parameters())

Akcja otwiera menu kontekstowe diagramu o nazwie *menu1*. Jednocześnie do menu przekazywany jest kompletny zestaw parametrów zdefiniowanych w aktywnym diagramie.

5.63 PI

Przeznaczenie

Zwraca wartość liczby pi.

Składnia

PI ()

Zobacz też

Funkcje *Sin*, *Cos*, *Tan*

5.64 Pow

Przeznaczenie

Funkcja wylicza wartość liczby podniesionej do podanej potęgi.

Składnia

Pow (*podstawa*, *potęga*)

Parametry

podstawa

Wartość, która ma być podniesiona do potęgi.

potęga

Wartość potęgi.

Zobacz też

Funkcje *Sqrt*

5.65 Property

Przeznaczenie

Funkcja zwraca wartość właściwości globalnej o podanej nazwie.

Składnia

Property (*nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości globalnej, którą należy pobrać.

Przykład

Property(g1)

Pobierana jest definicja właściwości globalnej o nazwie *g1.* Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

!g1

Zobacz też

Funkcje *LocalProperty*

5.66 RangeAggregate

Przeznaczenie

Funkcja zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu.

Składnia

RangeAggregate(*nazwa_agregatu, horyzont_początek,horyzont_koniec, okres_odświeżania*)

RangeAggregate(*nazwa_zmiennej, nazwa_agregatu, horyzont_początek, horyzont_koniec, okres_odświeżania*)

Parametry

nazwa_agregatu

Nazwa agregatu, który należy wyliczyć. Dostępne nazwy to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Nazwa *none* oznacza odczyt danych surowych. Opis agregatów znajduje się w dokumentacji modułu archiwizacji Aspad (Patrz: Asix.Evo_rodzaje_agregatów_danych_archiwalnych.PDF; Asix.PDF/CHM, rozdz. *7. Aspad - Program Archiwizacji Danych*).

horyzont_początek

Początek horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime*).

horyzont_koniec

Koniec horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime*).

okres_odświeżania

Okres ponawiania odczytu danych archiwalnych. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan*).

nazwa_zmiennej

Nazwa zmiennej, której wartości historyczne są pobierane. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej używana jest zmienna wynikająca z kontekstu użycia.

Przykład

RangeAggregate(min,”HOUR”,”NOW”,”00:00:10”)

Funkcja zwraca minimalną wartość zmiennej kontekstowej z okresu od początku godziny do chwili bieżącej. Wartość funkcji jest przeliczana co 10 sekund.

RangeAggregate(z1, average,”HOUR-1H”,”HOUR”,”00:01:00”)

Funkcja zwraca średnią wartość zmiennej *z1* z okresu ostatniej zakończonej godziny. Wartość funkcji jest przeliczana co 1 minutę.

Zobacz też

- Funkcje *HorizonAggregate*, *ToDateTime*, *OPCTime*, *ToTimeSpan*,

- Asix.Evo_rodzaje_agregatów_danych_archiwalnych.PDF

- Asix.PDF/CHM, rozdz. *7. Aspad - Program Archiwizacji Danych*

5.67 RateOfChange

Przeznaczenie

Funkcja wylicza prędkość zmian wartości dowolnego wyrażenia. Typowo jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

RateOfChange(*wartość, okres_przeliczania*)

Parametry

wartość

Wyrażenie, którego prędkość zmian wartości jest monitorowana.

okres_przeliczania

Okres czasu wyrażony w sekund określający, jak często wyliczana jest prędkość zmiany wartości monitorowanej.

Przykład

RateOfChange(Variable(s1),5)>20

Co 5 sekund wyliczana jest średnia prędkość zmiany wartości w ostatnim okresie obliczeniowym. Następnie sprawdzane jest, czy ta prędkość jest większa niż 20 jednostek/sekundę.

Zobacz też

- funkcje *DeadBand*, *DelayedState*

- Asix.Evo_System_alarmów.PDF/CHM, *5.2 Strategia warunkowa*

5.68 RawValue

Przeznaczenie

Funkcja zwraca bieżącą wartość surową zmiennej procesowej. Jest to oryginalna wartość zmiennej odczytana przez drajwer kanału komunikacyjnego przed zastosowaniem funkcji przeliczającej zdefiniowanej w bazie zmiennych w atrybucie *Funkcja przeliczająca odczytu*. Głównym zastosowaniem funkcji jest dostęp do wartości zmiennej w wyrażeniach użytych w

atrybucie *Funkcja przeliczająca odczytu* w celu przeliczenia wartości surowej zmiennej. Tak przeliczona wartość zmiennej jest potem dostępna poprzez funkcję *Variable*.

UWAGA: w przypadku korzystania z kanału komunikacyjnego *Asix6,* wartość zwracana przez funkcję jest już wstępnie przeliczona przez moduł komunikacyjny Asmen na podstawie atrybutu *Funkcja przeliczająca*.

Składnia

RawValue ()

RawValue (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość surową należy zwrócić. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej odczytywana jest wartość zmiennej wynikającej z kontekstu użycia. W przypadku użycia funkcji w wyrażeniu atrybutu *Funkcja przeliczająca odczytu*, zmienną kontekstową jest zmienna, której definicja atrybutu dotyczy.

Zobacz też

Funkcje *Variable*, *NewValue*

5.69 RelToAbsX

Przeznaczenie

Funkcja konwertuje współrzędną poziomą lub szerokość z wartości względnej wyrażonej w pikselach na wartość bezwzględną (0 do 1 000 000). Wartości bezwzględne są przechowywane w definicjach współrzędnych. Przejście na wartości względne wykonywane jest w trakcie wyświetlania obiektu i jest dostosowywane do aktualnych rozmiarów diagramu. Współrzędna bezwzględna 1 000 000 odpowiada prawej krawędzi diagramu. Funkcja używana jest w przypadku implementacji ruchu lub zmiany rozmiaru obiektów.

Składnia

RelToAbsX(*współrzędna_X*)

Parametry

współrzędna_X

Współrzędna X lub szerokość obiektu w jednostkach względnych (pikselach), którą należy przeliczyć na wartość bezwzględną.

Zobacz też

- funkcje *AbsToRelX, AbsToRelY, RelToAbsY*

- Asix.Evo_Techniki_budowy_diagramów.PDF/CHM, *13 Animacja ruchu i zmiana rozmiaru obiektów*

5.70 RelToAbsY

Przeznaczenie

Funkcja konwertuje współrzędną pionową lub wysokość z wartości względnej wyrażonej w pikselach na wartość bezwzględną (0 do 1 000 000). Ogólne zasady użycia są takie same jak w przypadku funkcji *RelToAbsX*.

Składnia

RelToAbsY(*współrzędna_Y*)

Parametry

współrzędna_Y

Współrzędna Y lub wysokość obiektu w jednostkach względnych(pikselach), którą należy przeliczyć na wartość bexwzględną.

Zobacz też

- funkcje *AbsToRelX, AbsToRelY, RelToAbsX*

- - Asix.Evo_Techniki_budowy_diagramów.PDF/CHM, *13 Animacja ruchu i zmiana rozmiaru obiektów*

5.71 Replace

Przeznaczenie

Funkcja tworzy napis poprzez zamianę wszystkich wystąpień wskazanego podciągu na inny ciąg znaków.

Składnia

Replace(*napis, stary_tekst, nowy_tekst*)

Parametry

napis

Napis, na podstawie którego będzie tworzony napis zmodyfikowany.

stary_tekst

Ciąg znaków, który należy zastąpić nową zawartością.

nowy_tekst

Nowy ciąg znaków, który zamieni poprzednią zawartość.

Zobacz też

Funkcje *Substring*, *Concat*

5.72 Round

Przeznaczenie

Funkcja zaokrągla liczbę zmiennoprzecinkową do podanej liczby cyfr ułamkowych.

Składnia

Round (*wartość, liczba_cyfr*)

Parametry

wartość

Wartość, która ma być zaokrąglona.

liczba_cyfr

Liczba cyfr ułamkowych w zwracanej wartości.

Przykład

Round (Variable(), 3)

Wywołanie zwraca wartość zmiennej kontekstowej zaokrągloną do 3 miejsc po przecinku. Jeżeli celem byłoby jedynie wyświetlenie zaokrąglonej wartości na ekranie, to można też użyć funkcji formatującej teksty *Format* - w poniższy sposób:

Format(”{0:f3}”,Variable())

Zobacz też

Funkcja *Format*

5.73 Sin

Przeznaczenie

Funkcja wylicza wartość funkcji sinus.

Składnia

Sin (*kąt*)

Parametry

kąt

Argument funkcji sinus wyrażony w radianach

Przykład

Sin(PI() * Variable(z1) / 180)

Funkcja liczy sinus kąta podanego w stopniach i pobranego z wartości zmiennej *z1*.

Zobacz też

Funkcje Cos, *Tan*, *Pi*

5.74 Sqrt

Przeznaczenie

Funkcja wylicza wartość pierwiastka kwadratowego.

Składnia

Sqrt (*argument*)

Parametry

argument

Argument funkcji pierwiastka kwadratowego.

Zobacz też

Funkcja *Pow*

5.75 Substring

Przeznaczenie

Funkcja zwraca napis będący fragmentem napisu podanego w wywołaniu funkcji.

Składnia

Substring(*napis, początek*)

Substring(*napis, początek, długość*)

Parametry

napis

Napis, z którego zostanie wycięty fragment.

początek

Indeks pierwszego znaku wycinanego fragmentu. Pierwszy znak ma numer 0.

długość

Ilość wycinanych znaków. Brak parametru oznacza wycięcie wszystkich znaków aż do końca napisu.

Przykład

Substring(Attribute(Description), 0, 10)+"..."

Wyrażenie zwraca pierwsze 10 znaków opisu zmiennej kontekstowej uzupełnione o znaki ….

Zobacz też

Funkcje *Replace*, *Concat*

5.76 TableElement

Przeznaczenie

Funkcja zwraca wskazany element z wartości tablicowej. Funkcja przeznaczona głównie do uzyskania dostępu do elementów tablicowych zmiennych procesowych w wyrażeniach użytych we właściwościach obiektów.

Składnia

TableElement(*tablica, indeks*)

Parametry

tablica

Wartość tablicowa, której element należy pobrać. Typowo jest to wartość tablicowej zmiennej procesowej. Można również użyć napis – w tym przypadku zwrócony jest pojedynczy znak.

Gdy tablica jest dwuwymiarowa, zwracana jest podtablica zawierająca wszystkie elementy w drugim wymiarze odpowiadające wskazanemu indeksowi w pierwszym wymiarze.

indeks

Indeks elementu, który należy zwrócić. Pierwszy element ma indeks równy 0.

Przykład

TableElement(Variable(),0)

Funkcja zwraca pierwszy element z wartości tablicowej zmiennej kontekstowej.

Zobacz też

funkcja *TableLength*

5.77 TableLength

Przeznaczenie

Funkcja zwraca liczbę elementów w wartości tablicowej.

Składnia

TableLength (*tablica*)

Parametry

tablica

Wartość tablicowa, której ilość elementów zostanie zwrócona. Typowo jest to wartość tablicowej zmiennej procesowej. Można również użyć napis – w tym przypadku zwrócona zostanie długość napisu.

Przykład

TableLength(Variable())

Funkcja zwraca ilość elementów w wartości zmiennej kontekstowej.

TableLength(Attribute(Name))

Zwraca długość nazwy zmiennej kontekstowej.

Zobacz też

funkcja *TableElement*

5.78 Tan

Przeznaczenie

Funkcja wylicza wartość funkcji tangens.

Składnia

Tan (*kąt*)

Parametry

kąt

Argument funkcji tangens wyrażony w radianach.

Zobacz też

Funkcje *Sin*, Cos, *Pi*

5.79 Text

Przeznaczenie

Funkcja zwraca napis z puli napisów wielojęzycznych. Zwracany jest wariant napisu odpowiedni dla aktualnego języka pracy. Użycie funkcji w definicji właściwości *Tekst* obiektu *Tekst* spowoduje, że w momencie zmiany języka obiekt automatycznie zmieni wyświetlany napis.

Składnia

Text (*identyfikator_napisu*)

Parametry

identyfikator_napisu

Identyfikator napisu wielojęzycznego zdefiniowanego w aplikacji.

Zobacz też

Asix.Evo_Parametryzacja_aplikacji, *4 Parametryzacja aplikacji wielojęzycznych*

5.80 ToAsix6Date

Przeznaczenie

Funkcja przelicza wartość typu data i czas (typ DateTime platformy .NET) lub tekstowy opis daty i czasu na liczbę sekund, która upłynęła od 1.1.1970. Zwracana wartość jest liczbą 32-bitową bez znaku. Funkcja używana głównie przy konwersji aplikacji Asix6.

Składnia

ToAsix6Time (*data_czas*)

Parametry

data_czas

Przeliczana data i czas, która może być:

- wartością typu data i czas (DateTime)
- tekstem, którego format jest zgodny z formatem stosowanym w funkcji *ToDateTime*

Zobacz też

Funkcje *ToDateTime*, *FromAsix6Date*, *OPCTime*

5.81 ToAsix6Time

Przeznaczenie

Funkcja przelicza wartość typu zakres czasu (typ TimeSpan platformy .NET) lub tekstowy opis okresu czasu na liczbę milisekund. Zwracana wartość jest liczbą 32-bitową bez znaku.

Składnia

ToAsix6Time (*zakres_czasu*)

Parametry

zakres_czasu

Przeliczany zakres czasu, który może być:

- wartością typu zakres czasu (TimeSpan)
- tekstem, którego format jest zgodny z formatem stosowanym w funkcji *ToTimeSpan*

Przykład

ToAsix6Time(”30:00”)

Uzyskana wartość jest równa ilości milisekund w 30-godzinnym okresie czasu.

Zobacz też

Funkcje *ToTimeSpan*, *FromAsix6Time*, *OPCTime*

5.82 ToBool

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na wartość logiczną *true/false*.

Składnia

ToBool (*wartość*)

Parametry

wartość

Konwertowana wartość. W przypadku wartości liczbowych, jeżeli wartość jest różna 0, to zwracana jest wartość *true*, w przeciwnym przypadku zwracane jest *false*. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Przykład

ToBool(Attribute(*wazna*))

Z bazy definicji zmiennych pobierana jest treść atrybutu *wazna* zmiennej kontekstowej. Następnie treść atrybutu jest konwertowana na liczbę i zamieniana na wartość logiczną. Dopuszczalne jest także bezpośrednie umieszczenie w atrybucie tekstów true lub false.

5.83 ToDateTime

Przeznaczenie

Funkcja konwertuje tekstowy opis daty i czasu na format liczbowy daty i czasu (typ DateTime platformy .NET)

Składnia

ToDateTime (*data_czas*)

Parametry

data_czas

Data i czas w formacie tekstowym. Dopuszczalny jest dowolny format rozpoznawalny przez system jako data i czas. Jeżeli podany jest tylko czas, to jako data zostanie przyjęty dzień bieżący.

Przykład

DateTime(”15:10”)

Uzyskana zostanie wartość równoważna godzinie 15:10 i dacie dnia bieżącego.

DateTime(Variable(z1))

Wartość zmiennej *z1* zostanie zwrócona w formacie daty i czasu. Wartość zmiennej musi być tekstem konwertowalnym na datę i czas.

Zobacz też

Funkcje *ToAsix6Date*, *FromAsix6Date*, *OPCTime*

5.84 ToDouble

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na liczbę zmiennoprzecinkową podwójnej precyzji.

Składnia

ToDouble (*wartość*)

Parametry

wartość

Konwertowana wartość.

Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik daty OLE Automation.

Przykład

ToDouble(Variable(i1))/Variable(i2)

Jeżeli zmienne *i1* I *i2* są całkowitoliczbowe, to dzielenie wartości tych zmiennych zwracało by także liczbę całkowitą, np. 4/8 = 0. Konwersja jednego z argumentów dzielenia na liczbę zmiennoprzecinkową powoduje, że wynik będzie dokładny, 4/8 = 0.5

Zobacz też

Funkcja *ToSingle*

5.85 ToFullPath

Przeznaczenie

Funkcja zwraca bezwzględną ścieżkę do pliku dla ścieżki podanej względem katalogu definicyjnego aplikacji.

Składnia

ToFullPath (*nazwa_pliku*)

Parametry

nazwa_pliku

Nazwa pliku lub katalogu, która zostanie uzupełniona z uwzględnieniem pełnej ścieżki katalogu aplikacji. Pusta nazwa pliku spowoduje zwrócenie ścieżki katalogu aplikacyjnego.

5.86 ToInt8

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 8-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt8 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje *ToInt16*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.87 ToInt16

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 16-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt16 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Przykład

ToInt16(Variable(z1)&0xffff)

Operacja & powoduje, że wartość zmiennej *z1* jest ograniczana do zakresu liczby 16-bitowej.

Zobacz też

Funkcje *ToInt8*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.88 ToInt32

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 32-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt32 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.89 ToInt64

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 64-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt64 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik typu *FileTime* systemu Windows.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt32*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.90 ToSingle

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na liczbę zmiennoprzecinkową pojedynczej precyzji.

Składnia

ToSingle (*wartość*)

Parametry

wartość

Konwertowana wartość.

Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcja *ToDouble*

5.91 ToString

Przeznaczenie

Funkcja konwertuje wartość liczbową interpretowaną jako sekwencję kodów ASCII na tekst Unicode.

Składnia

ToString (*wartość*)

Parametry

wartość

Konwertowana wartość liczbowa. Obsługiwane są także tablice wartości.

Przykład

ToString(Variable(msgtext))

Wartość zmiennej *msgtext* jest konwertowana na tekst. Zakładając, że ta wartość (typu 32-bitowa liczba całkowita) była równa 0x414243, to zwrócony zostanie tekst „CAB”. Kolejność znaków wynika z fizycznego ułożenia kolejnych bajtów liczby w pamięci – młodsze bajty występują jako pierwsze.

5.92 ToTimeSpan

Przeznaczenie

Funkcja konwertuje tekstowy opis zakresu czasu na format zakresu czasu (typ TimeSpan platformy .NET)

Składnia

ToTimeSpan (*zakres*)

Parametry

zakres

Zakres czasu przekazywany jest w formacie tekstowym. Obsługiwane są poniższe konwencje:

- d.h:m:s h:m:s h:m , gdzie d, h, m, s są liczbami nieujemnymi oznaczającymi kolejno dni, godziny, minuty i sekundy, np.: "3.10:5:4" lub "123:00"
- ciąg złożony ze składników ze zbioru (nd, nD, nh, nH, nm, nM, ns, nS), gdzie n jest liczbą nieujemną, a d, h, m, s oznaczają kolejno dni, godziny, minuty i sekundy, np.: "20d" lub "3D 4H 2M 1S" lub "15M10s"

Zobacz też

Funkcje *ToAsix6Time*, *FromAsix6Time*

5.93 ToUInt8

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 8-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt8 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.94 ToUInt16

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 16-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt16 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*, *ToUInt64*

5.95 ToUInt32

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 32-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt32 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt64*

5.96 ToUInt64

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 64-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt64 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik typu *FileTime* systemu Windows.

Zobacz też

Funkcje *ToInt8*, *ToInt16*, *ToInt32*, *ToInt64*, *ToUInt8*, *ToUInt16*, *ToUInt32*

5.97 ValueOrDefault

Przeznaczenie

Funkcja zwraca wartość przekazaną w parametrze wywołania lub wartość domyślną (zastępczą) w przypadku, gdy wartość podstawowa jest niezdefiniowana lub niepoprawna. Funkcja pozwala uprościć niektóre wyrażenia wykorzystujące konstrukcje warunkowe.

Składnia

ValueOrDefault (*wartość, wartość_domyślna*)

Parametry

wartość

Wartość podstawowa, zwracana przez funkcję, gdy jest różna od null (jest ustawiona) i w przypadku liczby zmiennoprzecinkowej jest liczbą poprawną.

wartość_domyślna

Wartość zastępcza zwracana, gdy pierwszy argument wywołania funkcji jest równy null (nieustawiony) lub jest niepoprawną liczbą zmiennoprzecinkową.

Przykład

ValueOrDefault(Variable(v1), -1)

Wyrażenie zwraca wartość zmiennej v1 jeżeli została ona już odczytana. W przeciwnym wypadku zwraca wartość -1. Powyższe wyrażenie jest równoważne wyrażeniu warunkowemu *Variable(v1)!=null ? Variable(v1) : -1*

5.98 VarChanged

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o zmianie wartości zmiennej procesowej w ostatnim okresie czasu. Funkcja ma dwa przeznaczenia. Jeżeli wartość okresu sprawdzania jest większa od 0, to funkcja jest typowo używana w warunkach stanów obiektów (lub wyrażeniach warunkowych) – służy do sygnalizowania na diagramie faktu niedawnej zmiany wartości. Jeżeli okres sprawdzania jest równy 0, to funkcja używana jest w wyrażeniach warunków terminarza akcji. Pozwala to na uruchamianie akcji operatorskiej w reakcji na każdorazową zmianę wartości zmiennej.

Składnia

VarChanged (*okres*)

VarChanged (*nazwa_zmiennej, okres*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, dla której wyliczany jest status zmiany wartości. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, kontrolowana jest wartość zmiennej wynikającej z kontekstu użycia.

okres

Okres czasu sprawdzania zmian wartości podany w sekundach.

Zobacz też

Funkcja *VarTime*, Terminarz

5.99 Variable

Przeznaczenie

Funkcja zwraca bieżącą wartość zmiennej procesowej po zastosowaniu wszystkich przeliczeń wynikających z definicji zmiennej w bazie definicji zmiennych. Dostęp do wartości nieprzeliczonej dostępny jest poprzez funkcję *RawValue*.

Składnia

Variable ()

Variable (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość należy zwrócić. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, odczytywana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

Variable(“#_lmt”)

Odczytywana jest wartość zmiennej procesowej o nazwie zbudowanej przez dodanie sufiksu *_lmt* do nazwy zmiennej wynikającej z kontekstu użycia. Jeżeli wyrażenie jest użyte w

miejscu, które będzie wymagało przejścia na postać tekstową, to konwersja zostanie wykonana w sposób domyślny dla typu wartości. Jeżeli wymagana jest kontrola formatowania, to należy stosować funkcje *VarStringValue* lub *Format*.

Zobacz też

Funkcje *VarIsGood*, *VarIsNotGood*, *VarStatus*, *VarTime*, *RawValue*, *VarStringValue*, *Format*

5.100 VarIsGood

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy status wartości zmiennej jest poprawny. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarIsGood ()

VarIsGood (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status poprawności jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, sprawdzany jest status zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *VarIsNotGood*, *VarStatus*, *VarTime*

5.101 VarIsNotGood

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy status wartości zmiennej jest niepoprawny - czyli aktualna wartość zmiennej jest niewiarygodna. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarIsNotGood ()

VarIsNotGood (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status poprawności jest zwracany. W przypadku wariantu funkcji bez podania nazwy zmiennej, sprawdzany jest status zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *VaIsGood*, *VarStatus*, *VarTime*

5.102 VarStatus

Przeznaczenie

Funkcja zwraca pełny numeryczny status wartości zmiennej zgodny ze standardem OPC. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarStatus ()

VarStatus (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status wartości jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, zwracany jest status zmiennej wynikającej z kontekstu użycia.

Przykład

(VarStatus()&0xc0)==0xc0

Działanie powyższego wywołania jest identyczne z użyciem funkcji *VarIsGood()*

Zobacz też

Funkcje *VarIsGood*, *VarIsNotGood*, *VarTime*

5.103 VarStringValue

Przeznaczenie

Funkcja zwraca wartość tekstową zmiennej procesowej utworzoną na podstawie formatu zdefiniowanego w bazie definicji zmiennych. Alternatywną metodą formatowania wartości jest funkcja *Format*.

Składnia

VarStringValue ()

VarStringValue (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość należy zwrócić w formacie tekstowym. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, odczytywana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

Concat (VarStringValue(), ” ”, Attribute(Unit))

Zwracany jest napis złożony ze sformatowanej wartości zmiennej procesowej i nazwy jednostki pobranej z bazy definicji zmiennych.

VarStringValue(z1)

Zwraca wartość zmiennej *z1* w tekstowej formie. Jeżeli nie jest wymagana dodatkowa funkcjonalność, to w definicji właściwości obiektu identyczny efekt można uzyskać poprzez skrótową notację:

&z1

Zobacz też

Funkcje *Variable*, *Format*

5.104 VarTime

Przeznaczenie

Funkcja zwraca czas ostatniej aktualizacji wartości zmiennej procesowej. W przypadku typowych kanałów komunikacyjnych jest to czas ostatniego odświeżenia. Czas zmiennej może się zmienić nawet wtedy, gdy nie zmienia się wartość zmiennej. W przypadku kanału wirtualnego czas aktualizacji jest czasem ostatniego zapisu zmiennej.

Składnia

VarTime ()

VarTime (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której czas aktualizacji wartości jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej zwracany jest czas aktualizacji zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *VaIsGood*, *VarIsNotGood*, *VarStatus*, *Variable*

5.105 XOnScreen

Przeznaczenie

Funkcja przelicza współrzędną poziomą z wartości względnej dla wskazanego monitora na wartość bezwzględną dla pełnego pulpitu. Funkcja przeznaczona do użycia w akcjach *OpenWindow* i *OpenDiagram*. Pozwala na pozycjonowanie otwieranych okien na wybranych monitorach bez konieczności znajomości dokładnej konfiguracji systemu.

Składnia

XOnScreen(*współrzędna_X*, *monitor*)

Parametry

współrzędna_X

Wartość względnej koordynaty poziomej.

monitor

Numer monitora, względem którego nastąpi przeliczenie koordynaty względnej *współrzędna_X* na koordynatę bezwzględną pulpitu. Wartość 0 oznacza użycie monitora, na którym znajduje się kursor myszki. Wartości większe od 0 to numery monitorów zgodne z numeracją systemu operacyjnego.

Przykład

OpenDiagram (diag1, pokoj=11, null, $Dialog, XOnScreen(0,2), YOnScreen(0,2), “NoTitleBar,FixedSize”, $None)

Akcja spowoduje otwarcie okna diagramu w lewym górnym narożniku drugiego monitora.

Zobacz też

- funkcja *YOnScreen*

- Asix.Evo_Akcje.PDF/CHM, akcje *OpenDiagram*, *OpenWindow*

5.106 XorColor

Przeznaczenie

Funkcja zwraca dopełnienie (XOR) podanego koloru.

Składnia

XorColor (*kolor*)

Parametry

kolor

Parametr koloru może być wartością numeryczną pobraną z właściwości obiektu lub uzyskaną przy pomocy funkcji Color. Kolor można podać też w postaci tekstowej, jako nazwę koloru lub wartości liczbowe składowych rozdzielone znakami średnika.

Przykład

XorColor (red)

Wywołanie zwraca dopełnienie koloru czerwonego.

XorColor(LocalProperty("Color"))

Wywołanie zwraca dopełnienie koloru pobranego z właściwości *Color* obiektu.

Zobacz też

Funkcja *Color*

5.107 YOnScreen

Przeznaczenie

Funkcja przelicza współrzędną pionową z wartości względnej dla wskazanego monitora na wartość bezwzględną dla pełnego pulpitu. Ogólne zasady użycia są identyczne jak dla funkcji *XOnScreen.*

Składnia

YOnScreen(*współrzędna_Y, monitor*)

Parametry

współrzędna_Y

Wartość względnej koordynaty pionowej.

monitor

Numer monitora, względem którego nastąpi przeliczenie koordynaty względnej *współrzędna_Y* na koordynatę bezwzględną pulpitu. Wartość 0 oznacza użycie monitora, na którym znajduje się kursor myszki. Wartości większe od 0 to numery monitorów zgodne z numeracją systemu operacyjnego.

Zobacz też

- funkcja *XOnScreen*

- Asix.Evo_Akcje.PDF/CHM, akcje *OpenDiagram*, *OpenWindow*