Wonderware® FactorySuite™ Справочное руководство по InTouch

Руководство пользователя

Редакция B
Май 1998

Корпорация Wonderware

Все права сохранены. Никакая часть настоящей документации не может воспроизводиться, храниться в информационно системе или передаваться любым способом, электронным или механическим, или путем фотокопирования, записи или как-то иначе без предварительного письменного согласия корпорации Wonderware Corporation. Использование содержащейся здесь информации не влечет за собой никакой ответственности, связанной с авторскими или патентными правами. Хотя при подготовке этой документации авторами и издателями были приложены все усилия, они не несут ответственность за возможные ошибки или неточности, равно как за возможный ущерб, причиненный в результате использования содержащейся здесь информации.

Содержащаяся здесь информация может быть изменена без предупреждения и не носит характера обязательств со стороны корпорации Wonderware. Описанное в документации программное обеспечение предоставляется по соглашениям о лицензии и неразглашении. Это программное обеспечение может использоваться только на условиях данных соглашений.

Данное руководство переведено с английской версии B от 5/98.

© 1998 Wonderware Corporation. Все права сохранены.
100 Technology Drive
Irvine, CA 92618
U.S.A.
(949) 727-3200
http://www.wonderware.com

Торговые марки

Все упоминаемые в этой книге названия, известные как торговые или сервисные марки, помечены должным образом. Корпорация Wonderware не может гарантировать точность этой информации. Использование любых названий в этой книге не следует считать нарушением каких-либо торговых или сервисных марок.

Wonderware – зарегистрированная торговая марка Wonderware Corporation.

Wonderware FactorySuite, InTouch, WindowMaker, WindowViewer, SQL Access Manager, Recipe Manager, SPC Pro, DBDump, DBLoadd, HDMerge, HistData, Wonderware Logger, InControl, InTrack, InBatch, IndustrialSQL, FactoryOffice, Scout, SuiteLink и NetDDE – торговые марки Wonderware Corporation.

Оглавление

Введение ... xi

Глава 1 - Системные тэги ... 1-1
$AccessLevel безопасность ... 1-2
$AlarmLogging алармы ... 1-3
$AlarmPrinterError алармы ... 1-3
$AlarmPrinterNoPaper алармы ... 1-4
$AlarmPrinterOffline алармы ... 1-4
$AlarmPrinterOverflow алармы ... 1-5
$ApplicationChanged приложение ... 1-5
$ApplicationVersion приложение ... 1-6
$ChangePassword безопасность ... 1-6
$ConfigureUsers безопасность ... 1-7
$Date система ... 1-7
$DateString система ... 1-8
$DateTime система ... 1-8
$Day система ... 1-8
$HistoricalLogging архивные ... 1-9
$Hour система ... 1-9
$InactivityTimeout безопасность ... 1-10
$InactivityWarning безопасность ... 1-10
$LogicRunning система ... 1-11
$Minute система ... 1-11
$Month система ... 1-11
$Msec система ... 1-12
$NewAlarm алармы ... 1-12
$ObjHor система ... 1-12
$ObjVer система ... 1-13
$Operator безопасность ... 1-13
$OperatorEntered безопасность ... 1-14
$PasswordEntered безопасность ... 1-14
$Second система ... 1-15
$StartDdeConversations система ... 1-15
$System алармы ... 1-15
$Time система ... 1-16
$TimeString система ... 1-16
$Year система ... 1-16

Глава 2 - Типы тэгов ... 2-1
Внутренние тэги (Memory) ... 2-2
Внешние тэги (I/O) ... 2-3
Косвенные тэги Indirect Discrete, Indirect Analog, Indirect Message ... 2-3
Прочие типы тэгов ... 2-4
Таблица соответствия типов тэгов и полей тэгов ... 2-4
.Ack алармы ... 2-7

.Alarm алармы 2-8
.AlarmDevDeadband алармы 2-9
.AlarmEnabled алармы 2-10
.AlarmValDeadband алармы 2-11
.ChartLength архив 2-12
.ChartStart архив 2-13
.Comment тэги 2-13
.DevTarget алармы 2-14
.DisplayMode архив 2-14
.EngUnits тэги 2-15
.HiHiLimit алармы 2-15
.HiHiStatus алармы 2-16
.HiLimit алармы 2-17
.HiStatus алармы 2-17
.LoLimit алармы 2-18
.LoLoLimit алармы 2-18
.LoLoStatus алармы 2-19
.LoStatus алармы 2-20
.MajorDevPct алармы 2-20
.MajorDevStatus алармы 2-21
.MaxEU тэги 2-22
.MaxRange архив 2-23
.MaxRaw тэги 2-24
.MinEU тэги 2-25
.MinorDevPct алармы 2-26
.MinorDevStatus алармы 2-27
.MinRange архив 2-28
.MinRaw тэги 2-29
.Name тэги 2-29
.Normal алармы 2-31
.OffMsg тэги 2-32
.OnMsg тэги 2-32
.Pen1 - .Pen8 архив 2-33
.Quality тэги 2-36
.QualityLimit тэги 2-39
.QualityLimitString тэги 2-39
.QualityStatus тэги 2-40
.QualityStatusString тэги 2-40
.QualitySubstatus тэги 2-41
.QualitySubstatusString тэги 2-42
.RawValue тэги 2-42
.Reference тэги 2-43
.ReferenceComplete тэги 2-43
.ROCPct алармы 2-44
.ROCStatus алармы 2-45
.ScooterLockLeft архив 2-46
.ScooterLockRight архив 2-47
.ScooterPosLeft архив 2-48
.ScooterPosRight архив 2-49
.SPCStatus SPC 2-50

.TagID тэги ... 2-50
.TimeDate тэги ... 2-51
.TimeDateString тэги ... 2-51
.TimeDateTime тэги ... 2-51
.TimeDay тэги ... 2-52
.TimeHour тэги ... 2-52
.TimeMinute тэги ... 2-52
.TimeMonth тэги ... 2-53
.TimeMsec тэги ... 2-53
.TimeSecond тэги ... 2-53
.TimeTime тэги ... 2-54
.TimeTimeString тэги ... 2-54
.TimeYear тэги ... 2-54
.Unack алармы ... 2-55
.UpdateCount архив ... 2-56
.UpdateInProgress архив ... 2-57
.UpdateTrend архив ... 2-58
.Value тэги ... 2-58
.AlarmGroup распределенные алармы ... 2-59
.NextPage распределенные алармы ... 2-59
.NumAlarms распределенные алармы ... 2-60
.PageNum распределенные алармы ... 2-60
.PrevPage распределенные алармы ... 2-61
.PriFrom распределенные алармы ... 2-61
.PriTo распределенные алармы ... 2-62
.ProviderReq распределенные алармы ... 2-62
.ProviderRet распределенные алармы ... 2-63
.QueryState распределенные алармы ... 2-63
.QueryType распределенные алармы ... 2-64
.Successful распределенные алармы ... 2-64
.TotalPages распределенные алармы ... 2-65
.Caption элементы управления окна ... 2-65
.Enabled элементы управления окна ... 2-66
.ListCount элементы управления окна ... 2-66
.ListIndex элементы управления окна ... 2-67
.NewIndex элементы управления окна ... 2-67
.ReadOnly элементы управления окна ... 2-68
.TopIndex элементы управления окна ... 2-68
.Value элементы управления окна ... 2-69
.Visible элементы управления окна ... 2-70

Глава 3 - Функции Quick-сценариев ... 3-1
Abs() математическая ... 3-2
Ack() алармы ... 3-2
ActivateApp() системная ... 3-2
almAckAll() распределенные алармы ... 3-3
almAckDisplay() распределенные алармы ... 3-3
almAckRecent() распределенные алармы ... 3-3
almAckSelect() распределенные алармы ... 3-5
almDefQuery() распределенные алармы ... 3-5

almMoveWindow() распределенные алармы ... 3-6
almQuery() распределенные алармы ... 3-7
almSelectAll() распределенные алармы ... 3-7
almSelectItem() распределенные алармы ... 3-8
almShowStats() распределенные алармы ... 3-8
ArcCos() математическая ... 3-8
ArcSin() математическая ... 3-9
ArcTan() математическая ... 3-9
ChangePassword() безопасность ... 3-10
Cos() математическая ... 3-10
DialogStringEntry() прочие ... 3-11
DialogValueEntry() прочие ... 3-12
DText() строка ... 3-13
Exp() математическая ... 3-14
FileCopy() системная ... 3-14
FileDelete() системная ... 3-15
FileMove() системная ... 3-16
FileReadFields() системная ... 3-18
FileReadMessage() системная ... 3-19
FileWriteFields() системная ... 3-20
FileWriteMessage() системная ... 3-21
GetNodeName() системная ... 3-21
GetPropertyD() GOT ... 3-22
GetPropertyI() GOT ... 3-22
GetPropertyM() GOT ... 3-23
Hide прочие ... 3-23
HideSelf прочие ... 3-23
HTGetLastError() архивные ... 3-24
HTGetPenName() архивные ... 3-24
HTGetTimeAtScooter() архивные ... 3-25
HTGetTimeStringAtScooter() архивные ... 3-25
HTGetValue() архивные ... 3-26
HTGetValueAtScooter() архивные ... 3-27
HTGetValueAtZone() архивные ... 3-28
HTScrollLeft() архивные ... 3-29
HTScrollRight() архивные ... 3-29
HTSelectTag() прочие ... 3-30
HTSetPenName() архивные ... 3-30
HTUpdateToCurrentTime() архивные ... 3-31
HTZoomIn() архивные ... 3-31
HTZoomOut() архивные ... 3-33
InfoAppActive() системная ... 3-34
InfoAppTitle() системная ... 3-34
InfoDisk() системная ... 3-35
InfoFile() системная ... 3-36
InfoInTouchAppDir() системная ... 3-37
InfoResources() системная ... 3-37
Int() математическая ... 3-38
IOSetAccessName() прочие ... 3-39
IOSetItem() прочие ... 3-40

IsAnyAsynchFunctionBusy() системная ... 3-41
Log() математическая ... 3-41
LogMessage прочие ... 3-42
LogN() математическая ... 3-42
Pi() математическая ... 3-42
PlaySound() прочие ... 3-43
PrintHT() архивные ... 3-43
PrintWindow() прочие ... 3-45
RecipeDelete() рецепты ... 3-46
RecipeGetMessage() рецепты ... 3-47
RecipeLoad() рецепты ... 3-48
RecipeSave() рецепты ... 3-49
RecipeSelectNextRecipe() рецепты ... 3-50
RecipeSelectPreviousRecipe() рецепты ... 3-51
RecipeSelectRecipe() рецепты ... 3-52
RecipeSelectUnit() рецепты ... 3-53
RestartWindowViewer системная ... 3-54
Round() математическая ... 3-54
SendKeys прочие ... 3-55
SetDDEAppTopic() прочие ... 3-56
SetDDEItem() прочие ... 3-56
SetPropertyD() GOT ... 3-57
SetPropertyI() GOT ... 3-57
SetPropertyM() GOT ... 3-58
Sgn() математическая ... 3-58
Show прочие ... 3-59
ShowAt() прочие ... 3-59
ShowHome прочие ... 3-60
ShowTopLeftAt() прочие ... 3-60
Sin() математическая ... 3-60
SPCConnect() SPC ... 3-61
SPCDisconnect() SPC ... 3-61
SPCDisplayData() SPC ... 3-61
SPCLocateScooter() SPC ... 3-62
SPCMoveScooter() SPC ... 3-62
SPCSaveSample() SPC ... 3-62
SPCSelectDataset() SPC ... 3-63
SPCSelectProduct() SPC ... 3-63
SPCSetControlLimits() SPC ... 3-63
SPCSetMeasurement() SPC ... 3-64
SPCSetProductCollected() SPC ... 3-64
SPCSetProductDisplayed() SPC ... 3-64
SPCSetRangeLimits() SPC ... 3-65
SPCSetSpecLimits() SPC ... 3-65
SQLAppendStatement() SQL ... 3-65
SQLClearParam() SQL ... 3-67
SQLClearStatement() SQL ... 3-67
SQLClearTable() SQL ... 3-67
SQLCommit() SQL ... 3-68
SQLConnect() SQL ... 3-69

SQLCreateTable() SQL ... 3-70
SQLDelete() SQL ... 3-71
SQLDisconnect() SQL ... 3-71
SQLDropTable() SQL ... 3-73
SQLEnd() SQL ... 3-73
SQLErrorMsg() SQL ... 3-73
SQLExecute() SQL ... 3-75
SQLFirst() SQL ... 3-75
SQLGetRecord() SQL ... 3-75
SQLInsert() SQL ... 3-76
SQLInsertEnd() SQL ... 3-76
SQLInsertExecute() SQL ... 3-78
SQLInsertPrepare() SQL ... 3-78
SQLLast() SQL ... 3-78
SQLLoadStatement() SQL ... 3-79
SQLManageDSN() SQL ... 3-79
SQLNext() SQL ... 3-80
SQLNumRows() SQL ... 3-80
SQLPrepareStatement() SQL ... 3-80
SQLPrev() SQL ... 3-82
SQLRollback() SQL ... 3-82
SQLSelect() SQL ... 3-83
SQLSetParamChar() SQL ... 3-86
SQLSetParamDate() SQL ... 3-86
SQLSetParamDateTime() SQL ... 3-87
SQLSetParamDecimal() SQL ... 3-87
SQLSetParamFloat() SQL ... 3-88
SQLSetParamInt() SQL ... 3-88
SQLSetParamLong() SQL ... 3-88
SQLSetParamNull() SQL ... 3-89
SQLSetParamTime() SQL ... 3-90
SQLSetStatement() SQL ... 3-90
SQLTransact() SQL ... 3-92
SQLUpdate() SQL ... 3-93
SQLUpdateCurrent() SQL ... 3-94
Sqrt() математическая ... 3-94
StartApp системная ... 3-94
StringASCII() строка ... 3-95
StringChar() строка ... 3-95
StringFromIntg() строка ... 3-96
StringFromReal() строка ... 3-96
StringFromTime() строка ... 3-97
StringInString() строка ... 3-98
StringLeft() строка ... 3-98
StringLen() строка ... 3-99
StringLower() строка ... 3-99
StringMid() строка ... 3-100
StringReplace() строка ... 3-101
StringRight() строка ... 3-103
StringSpace() строка ... 3-103

StringTest() строка 3-104
StringToIntg() строка 3-105
StringToReal() строка 3-105
StringTrim() строка 3-106
StringUpper() строка 3-106
Tan() математическая 3-107
Text() строка 3-107
Trunc() математическая 3-108
wcAddItem() элементы управления окна 3-108
wcClear() элементы управления окна 3-109
wcDeleteItem() элементы управления окна 3-109
wcDeleteSelection() элементы управления окна 3-110
wcErrorMessage() элемент управления окна 3-110
wcFindItem() элементы управления окна 3-111
wcGetItem() элементы управления окна 3-111
wcGetItemData() элементы управления окна 3-112
wcInsertItem() элементы управления окна 3-113
wcLoadList() элементы управления окна 3-113
wcLoadText() элементы управления окна 3-115
wcSaveList() элементы управления окна 3-116
wcSaveText() элементы управления окна 3-117
wcSetItemData() элементы управления окна 3-118
WWControl() прочие 3-119
WWExecute() WWDDE 3-120
WWPoke() WWDDE 3-121
WWRequest() WWDDE 3-122

Приложение -
Отладка функций Quick-сценариев A-1
Сообщения об ошибках элементов управления окна
и распределенных алармов A-2
Вывод сообщений с кодами ошибок A-5
Имена элементов SPC DDE A-5
Элементы управления и вывода на экран SPC DDE A-5
Элементы текущей выборки SPC DDE A-8
Элементы ручного ввода SPC DDE A-12
Элементы выбора SPC DDE A-13
Отладка функций SQL в сценариях A-16
Сообщения об ошибках с кодом результата A-16
Коды ошибок конкретных баз данных A-18

Предметный указатель I-1

Введение

Настоящее руководство содержит в алфавитном порядке справочную информацию по всем полям тэгов, свойствам элементов управления окнами, свойствам объектов алармов, системным тэгам и функциям сценариев InTouch. В нем также описаны функции дополнительных программ Recipe Manager, SPC Pro и SQL Access Manager. Ниже приводится краткое содержание трех глав этой книги:

Глава 1 — "Системные тэги" содержит описание всех предопределенных системных тэгов InTouch.

Глава 2 — "Поля тэгов" описывает все поля тэгов, относящиеся к простым и распределенным алармам, элементам управления окнами, архивным данным и тэгам.

Глава 3 — "Функции сценариев InTouch" описывает все встроенные в InTouch строковые, математические, системные, архивные функции, относящиеся к элементам управления Windows, распределенным алармам и дополнительным программам (Recipe, SPC Pro, SQL), а также прочие функции.

Условные обозначения

Пример обозначения	Описание
.ChartStart	В целом, полужирным шрифтом выделены обозначения .поля, системного тэга или функции.
Tagname	В примерах синтаксиса *курсивом* обозначены "заглушки" (места, где требуется ввод конкретной информации).
{ **.HiLimit | .HiHiLimit** }	В примерах синтаксиса фигурные скобки с вертикальной чертой между ними обозначают варианты возможного ввода.
[ErrorMessage=]	В квадратные скобки заключено то, что не является в синтаксисе обязательным.

Справа от названия каждого системного тэга, свойства, поля с точкой или функции сценария указывается категория использования (безопасность, алармы, приложение и т. д.). Например:

$AccessLevel **безопасность**

Об этом руководстве

В руководстве встречаются следующие пиктограммы, облегчающие поиск нужной информации по всей электронной документации.

Если при просмотре электронной версии данного руководства вам встретилась подобная ссылка, то это — «переход» к другому разделу. Нажав на него, вы окажетесь в указанном разделе или главе документации. После перехода в другой раздел у вас всегда имеется возможность перейти «назад» к исходному разделу.

Такой тип перекрестных ссылок указывает, что дополнительная информация содержится в другой книге FactorySuite.

Такие "подсказки" указывают более легкий или быстрый путь выполнения функции или задачи.

«Руководство пользователя InTouch» поможет ознакомиться со средой разработки и инструментальными средствами WindowMaker.

О том, как работать с окнами, графическими объектами, мастерами, элементами ActiveX и т. п., рассказывается в главе 2, "Работа с WindowMaker".

Подробное описание среды выполнения InTouch (WindowViewer) содержится в книге «*Руководство пользователя InTouch Runtime*».

«Руководство администратора FactorySuite» содержит исчерпывающую информацию об общих компонентах FactorySuite и требованиях к системе, советы по работе в сети, интеграции продуктов, сведения о технической поддержке и многое другое.

В пакет программ FactorySuite входит электронная документация по всем компонентам, включенным в пакет. Например, FactorySuite System Administrator's Guide, SPC Pro, SQLAccess Manager, Recipe Manager, IndustrialSQL Server, InControl и все 32-битные серверы ввода/вывода Wonderware. Если вы приобрели пакет FactorySuite Plus, то в нем также имеется электронная документация для компонентов InTrack и InBatch.

Требования к пользователю

Это руководство рассчитано на пользователя, который:

- Знаком с операционными системами Windows 95 и/или Windows NT.
- Умеет работать с мышью, меню Windows, выбирать параметры и пользоваться справочной системой.
- Имеет опыт программирования или знаком с одним из макроязыков. Необходимо также иметь представление об основных понятиях программирования, таких как переменные, выражения, функции и методы.

Техническая поддержка

Служба технической поддержки корпорации Wonderware оказывает различные виды помощи, связанные с любыми аспектами использования продуктов Wonderware.

Прежде чем обращаться в службу технической поддержки, попытайтесь найти в соответствующих руководствах InTouch возможные решения тех проблем, которые возникли у вас при работе с системой. Если же обращение в службу технической поддержки окажется необходимым, будьте готовы сообщить следующую информацию:

1. Серийный номер программного обеспечения.
2. Номер версии InTouch.
3. Тип и номер версии используемой операционной системы. Например, Microsoft Windows NT workstation версии 4.0.
4. Точный текст полученного сообщения об ошибке.
5. Любые, важные на ваш взгляд, листинги Wonderware Logger, утилиты Microsoft Diagnostic (MSD) или другой утилиты диагностики.
6. Описание предпринятых вами попыток по устранению проблемы и полученных результатов.
7. Инструкции по воспроизведению проблемы.
8. Номер, присвоенный вашему запросу службой технической поддержки Wonderware (если вы обращаетесь с этой проблемой не первый раз).

📖 Подробные сведения о службе технической поддержки даны в электронном «*Руководстве администратора системы FactorySuite*».

Просмотр лицензии FactorySuite

Информация о лицензии на систему FactorySuite может быть получена при помощи утилиты просмотра лицензии, которая запускается из диалогового окна InTouch Help **About**.

🖱 Для обращения к диалоговому окну **About** выберите команду **About** в меню **Help**.

📖 Дополнительная информация об этой утилите содержится в «*Руководстве администратора системы FactorySuite*».

Г Л А В А 1

Системные тэги

InTouch позволяет использовать несколько предопределенных системных тэгов. Эти тэги создаются автоматически и отличаются от других знаком доллара ($) в начале имени тэга. В среде выполнения InTouch обрабатывает значения этих тэгов в ответ на определенные события, происходящие в системе. Системные тэги могут использоваться везде, где используются любые другие тэги InTouch, например, со связями анимации в сценариях.

$AccessLevel безопасность

Определяет уровень доступа вошедшего в систему пользователя.

Использование `$AccessLevel`

Описание Это тэг только для чтения, чье значение определяется уровнем доступа (AccessLevel), назначенным для профиля текущего пользователя InTouch. Доступ к этому профилю осуществляется через пункт меню **Настройка пользователей** в программе WindowViewer.

Фактическое числовое значение **$AccessLevel** не важно для WindowViewer. Вся предполагаемая «защита доступа» должна быть создана при проектировании системы. Используя значение **$AccessLevel**, можно гибко варьировать доступ к различным ресурсам системы на основе спроектированной политике безопасности.

Тип данных Целый (только чтение)

Значения 0 - 9999

Примеры Данное выражение используется для связи видимости, чтобы сделать объект, такой как кнопка, видимым только для пользователя, имеющего необходимый уровень доступа:

```
$AccessLevel >= 2000;
Objects can have a "disable" link associated with them, with
the expression based on $AccessLevel.

$AccessLevel < 5411;

IF $AccessLevel <=500 THEN
   Show "Access Denied"; { всплывающее окно, запрещающее
доступ }

ELSE
   Show "Access Granted"; { всплывающее окно, разрешающее
доступ }

ENDIF;
```

См. также **$Operator, $OperatorEntered, $PasswordEntered; $ConfigureUsers**

$AlarmLogging алармы

Повторно инициализирует регистрацию и печать алармов в среде выполнения. Этот тэг равнозначен команде **Перезапустить регистрацию алармов** в меню **Специальные** WindowViewer.

Использование

```
$AlarmLogging = 1;
```

Описание

Установка значения 1 повторно инициализирует регистрацию алармов и печати в среде выполнения.

Замечание. Имя тэга $AlarmLogging нельзя использовать для отключения регистрации алармов или печати. Он используется исключительно для замены команды Window Viewer Special/Restart Alarm Log. Если вывести на дисплей значение $AlarmLogging, оно всгда будет равно 0 (отключено) - даже, если регистрация ведется. Установка $AlarmLogging в значение, не равное 1, не имеет смысла, и результат в этом случае не определен.

Тип данных Дискретный (только запись)

Значения 1

Пример Данное выражение равнозначно выполнению **Перезапустить регистрацию алармов** в меню **Специальные** WindowViewer.

```
$AlarmLogging = 1;
```

См. также **$HistoricalLogging**

$AlarmPrinterError алармы

Ошибка вывода аларма на печать.

Использование

```
$AlarmPrinterError
```

Описание Содержит значение 1, если есть ошибка при выводе алармов на печать.

Тип данных Дискретный (только чтение)

Значения 0 = Нет ошибки в принтере алармов
1 = Есть ошибка в принтере алармов

Пример Это выражение, применяемое в качестве описания анимационной связи типа «Вывод аналогового выражения», выдаст значение 1 при наличии ошибок при выводе алармов на печать, или 0 в противном случае.

```
$AlarmPrinterError

IF $AlarmPrinterError == 1 THEN
   DisplayMessageTag = "Ошибка в принтере алармов";

ELSE
   DisplayMessageTag = "Нет  ошибки в принтере алармов";

ENDIF;
```

$AlarmPrinterNoPaper алармы

В принтере алармов кончилась бумага.

Использование `$AlarmPrinterNoPaper`

Описание Содержит значение 1, если в принтере алармов нет бумаги.

Тип данных Дискретный (только чтение)

Значения 0 = В принтере алармов есть бумага
1 = В принтере алар¬ов нет бумаги

Пример Это выражение, применяемое при описании анимационной связи типа «Вывод аналогового выражения», выдаст значение 1, если в принтере для вывода алармов на печать нет бумаги, в противном случае 0.

```
$AlarmPrinterNoPaper
```

Использование в сценарии действия с условием OnTrue:

```
Óñëîâèå:
   $AlarmPrinterOffline == 1

Сценарий:
   show "window";
   playsound("c:\winnt\system32\alarm.wav",1);
```

$AlarmPrinterOffline алармы

Принтер алармов отключен.

Использование `$AlarmPrinterOffline`

Описание Содержит значение 1, если принтер отключен.

Тип данных Дискретный (только чтение)

Значения 0 = Принтер алармов включен

1 = Принтер алармов отключен

Пример Это выражение, применяемое при описании анимационной связи типа «Вывод аналогового выражения», выдаст значение 1, если принтер отключен, в противном случае 0.

```
$AlarmPrinterOffline

IF $AlarmPrinterOffline == 1 THEN
   Call PLCHorn( ); { Функция Quick-сценария, включающая
звуковой сигнал в цехе предприятия }

ENDIF;
```

$AlarmPrinterOverflow алармы

Буфер принтера алармов переполнен.

Использование `$AlarmPrinterOverflow`

Описание Содержит значение 1, если есть ошибка принтера алармов (На принтер послано слишком много символов).

Тип данных Дискретный (только чтение)

Значения 0 = Буфер принтера алармов не переполнен

1 = Буфер принтера алармов заполнен и начинает переполняться

Пример Это выражение, применяемое при описании анимационной связи типа «Вывод аналогового выражения», выдаст значение 1, если принтер переполнен, в противном случае 0.

```
$AlarmPrinterOverflow
```

$ApplicationChanged приложение

Сигнализирует о том, что изменилось ведущее приложение в архитектуре NAD (Network Application Development – Сетевая разработка приложений).

Использование `$ApplicationChanged`

Описание Значение этого тэга устанавливается в 1, когда с помощью меню **Сервис/Оповещение клиентов** в WindowMaker генерируется сигнал обновления. Применяется для распределенного обслуживания приложений.

Тип данных Действительный (только чтение)

Пример Если это выражение встречается в поле тэга сценария изменения данных, то оно вызывает на выполнение сценарий. При этом может быть выведено окно, информирующее пользователя о необходимости перезапустить WindowViewer для того, чтобы новые изменения в программе возымели эффект.

```
$ApplicationChanged
```

$ApplicationVersion приложение

Содержит номер текущей версии приложения. Этот номер меняется при каждом редактировании тэга или Quick-сценария в приложении.

Использование `$ApplicationVersion`

Описание Отсутствует

Тип данных Действительный (только чтение)

Пример При использовании этого выражения в качестве аналоговой анимационной связи типа «Вывод аналогового выражения», данный системный тэг выведет на дисплей текущую версию приложения, запущенного из WindowViewer.

```
$ApplicationVersion
```

$ChangePassword безопасность

Выводит на экран диалоговое окно для изменения пароля. Эквивалентно команде меню **Сервис/Безопасность/Смена пароля** в WindowViewer.

Использование `$ChangePassword`

Описание Установка значения в 1 вызывает вывод на дисплей диалогового окна для изменения пароля. Когда диалоговое окно закрывается, InTouch автоматически сбрасывает значение в 0. Установка этого системного тэга в значение, отличное от 1, не имеет смысла, и результат в этом случае будет не определен.

Тип данных Дискретный (только запись)

Значения 1

Пример Чтобы разрешить пользователю выводить на дисплей диалоговое окно для изменения пароля, можно создать дискретную кнопку. Эта кнопка должна иметь единственное связанное с ней действие, и это действие должно быть - «**Set**». При нажатии кнопка установит (присвоит единицу) системный тэг **$ChangePassword**, что вызовет появление на экране диалогового окна.

См. также **$AccessLevel, $OperatorEntered, $PasswordEntered, $Operator, $ConfigureUsers**

$ConfigureUsers безопасность

Выводит на экран общее диалоговое окно **Настройка пользователей**. Равнозначно команде **Безопасность** / **Смена пароля** в меню **Сервис** в WindowViewer.

Использование `$ConfigureUsers`

Описание Установка значения 1 вызывает вывод на дисплей диалогового окна конфигурирования пользователя. Когда диалоговое окно закрывается, InTouch автоматически сбрасывает значение в 0. Установка этого системного тэга в значение, отличное от 1, не имеет смысла и результат в этом случае будет не определен. Чтобы окно было выведено на дисплей, пользователь должен иметь уровень доступа $AccessLevel > 9000.

Тип данных Дискретный (только запись)

Значения 1

Пример Чтобы разрешить пользователю выводить на дисплей диалоговое окно для конфигурирования пользователей, можно создать дискретную кнопку. Эта кнопка должна иметь единственное связанное с ней действие, и это действие должно быть - «**Set**». При нажатии кнопка установит (присвоит единицу) системный тэг **$ConfigureUsers**, что вызовет появление диалогового окна на экране.

См. также **$Operator, $OperatorEntered, $ChangePassword, $PasswordEntered, $AccessLevel**

$Date система

Содержит целое число дней, прошедших со дня 1/1/70..

Использование `$Date`

Описание Нет

Тип данных Целый (только чтение)

Пример
```
StringFromTime(($Date*86400)+($Time/1000),3); { возвращает текущее время }
```

$DateString система

Содержит дату в формате, определенном в файле WIN.INI.

Использование `$DateString`

Описание Формат даты устанавливается в Панели управления Windows или путем двойного нажатия на поле времени в правом углу панели задач.

Тип данных Внутренний текстовый (только чтение)

Ïðèìåð

```
If StringRight($DateString,2)== "00" THEN
   LogMessage("The Year 2000!");
ENDIF;
```

$DateTime система

Содержит дробное число дней, прошедших со дня 1/1/70.

Использование `$DateTime`

Описание Отсутствует

Тип данных Действительный (только чтение)

Ïðèìåð

```
If StringFromTime($DateTime,4)== "Fri" THEN
   DisplayMessageTag = "It's Friday!";
ENDIF;
```

$Day система

Содержит текущий день месяца.

Использование `$Day`

Описание Значение с 1 до 31.

Тип данных Целый (только чтение)

Ïðèìåð

```
If $Day == 15 THEN
   Show "Mid-Month Washdown Window";
ENDIF;
```

$HistoricalLogging архивные

Следит и/или управляет за запуском и остановом регистрации архивных данных. Эквивалентно командам **Специальные/Перезапустить архивирование** или **Специальные/Остановить архивирование** в меню WindowViewer.

Использование `$HistoricalLogging`

Описание Установка этого системного тэга в 0 остановит регистрацию истории. Установка в 1 перезапустит регистрацию истории. Нельзя «запустить» регистрацию истории посредством **$HistoricalLogging**, если предварительно регистрация не была разрешена с помощью диалога WindowMaker **Сервис/Настройка/Архивирование**.

Тип данных Дискретный (чтение / запись)

Ïðèìåð

```
IF (InfoDisk("C",4,$Second)/1024)<50 THEN { остановить регистрацию архива, если на диске осталось меньше 50MB }
   $HistoricalLogging = 0;
ENDIF;
```

См. также **$AlarmLogging**

$Hour система

Содержит значение часа текущего времени суток.

Использование `$Hour`

Описание Значение от 0 до 23.

Тип данных Целый (только чтение)

Ïðèìåð Сценарий действия с условием On True:

```
$Hour == 20 AND $Second == 30
   PrintWindow("Day Batch Summary",1,1,0,0,0);
```

$InactivityTimeout безопасность

Указывает на то, что время, сконфигурированное на автоматический выход пользователя из системы, истекло.

Использование `$InactivityTimeout`

Описание Содержит значение 1, когда истекло время автоматического выхода пользователя из системы. Для установки этого времени следует выбрать меню **Сервис**/**Настройка**/**WindowViewer** или в Application Explorer под веткой **Настройка** дважды нажать на **WindowViewer**. Появится диалог **Свойства WindowViewer** с открытой вкладкой свойств **Общие**. Время вводится в секундах в группе параметров **Бездействие**.

Тип данных Дискретный (только чтение)

См. также **$InactivityWarning**

Ïðèìåð Ñöåíàðèé äåéñòâóåò ñ óñëîâèåì On True:

```
$InactivityTimeout == 1
Script:
   Show "You have been logged off window";
```

$InactivityWarning безопасность

Указывает на то, что время, сконфигурированное на вывод предупреждения о выходе пользователя из системы, истекло.

Использование `$InactivityWarning`

Описание Содержит значение 1, когда истекло время предупреждения об автоматическом выходе пользователя из системы. Для установки этого времени следует выбрать меню **Сервис**/**Настройка**/**WindowViewer** или в Application Explorer под веткой **Настройка** дважды нажать на **WindowViewer**. Появится диалог **Свойства WindowViewer** с открытой вкладкой свойств **Общие**. Время вводится в секундах в группе параметров **Бездействие**.

Тип данных Дискретный (только чтение)

Ïðèìåð

```
If $InactivityWarning == 1 THEN
   Show "You are about to be logged off-window";
ENDIF;
```

См. также **$InactivityTimeOut**

$LogicRunning система

Следит за и/или управляет выполнением сценариев. Эквивалентно командам **Программа/Начать программу** или **Программа/Остановить программу** в WindowViewer.

Примечание. Выполняемые асинхронные сценарии не могут быть остановлены. Однако можно запретить выполнение новых сценариев.

Использование `$LogicRunning`

Описание Установка значения в 1 запускает сценарий с логическими командами. Установка значения в 0 останавливает сценарий с логическими командами.

Тип данных Дискретный (чтение / запись)

$Minute система

Содержит значение минут текущего часа.

Использование `$Minute`

Описание Значение от 0 до 59.

Тип данных Целый (только чтение)

Ïðèìåð Введите следующее выражение для строки вывода значения связи анимации:

```
IF InfoFile("C:\InTouch.32\WIZ.INI",1,$Minute)==1 THEN
   LogMessage("The File Exists!");
ENDIF;
```

$Month система

Содержит текущий месяц года.

Использование `$Month`

Описание Значение от 1 до 12.

Тип данных Целый (только чтение)

Ïðèìåð

```
IF $Month==10 THEN
   CurrentMonthName = "October";
ENDIF;
```

$Msec система

Содержит текущее число миллисекунд. Частота обновления этого тэга устанавливается параметрами *Тактовый интервал* и *Обновление для переменных времени*. Выберите меню **Сервис**/**Настройка**/**WindowViewer** или в Application Explorer под веткой **Настройка** дважды нажмите на **WindowViewer**. Появится диалог **Свойства WindowViewer** с открытой вкладкой свойств **Общие**.

Использование `$Msec`

Описание Значение от 0 до 999.

Тип данных Целый (только чтение)

$NewAlarm алармы

Указывает на то, что возник новый аларм.

Использование `$NewAlarm`

Описание Содержит значение 1, когда возникает новый аларм. Этот тэг устанавливается только для нового локального аларма. Он не устанавливается при возникновении удаленных алармов.

Тип данных Дискретный (чтение / запись)

Пример Имя этого тэга можно связать с логической фугкцией **PlaySound**, которая выдает звуковой сигнал тревоги. Можно создать кнопку подтверждения, чтобы позволить оператору сбросить это значение в 0 и подтвердить аларм.

$ObjHor система

Содержит горизонтальную пиксельную координату центра выбранного объекта.

Использование `$ObjHor`

Описание Нет

Тип данных Целый (только чтение)

См. также **$ObjVer**

$ObjVer система

Содержит вертикальную пиксельную координату центра выбранного объекта.

Использование `$ObjVer`

Описание Отсутствует

Тип данных Целый (только чтение)

См. также **$ObjHor**

$Operator безопасность

Управляет способностью пользователя выполнять те или иные функции.

Использование `$Operator`

Описание Содержит имя вошедшего в систему пользователя.

Тип данных Текстовый (только чтение)

Пример Следующий сценарий позволяет управлять доступом к конкретному окну:

```
IF $Operator == "DayShift" THEN
   Show "Control Panel Window";
   ELSE Show "Wrong Operator";

ENDIF;
```

См. также **$OperatorEntered, $AccessLevel, $PasswordEntered, $ChangePassword, $ConfigureUsers**

$OperatorEntered безопасность

Используется для ввода допустимого имени пользователя.

Использование `$OperatorEntered`

Описание Создает окно входа пользователя в систему. С этим тэгом можно связать входные объекты касания и/или сценарии, чтобы пользователь мог вводить свое "User Name".

Примечание. Когда **$OperatorEntered** имеет допустимое значение, **$AccessLevel** и **$Operator** получают соответствующие предопределенные значения.

Тип данных Текстовый (чтение / запись)

См. также **$AccessLevel, $Operator, $PasswordEntered, $ChangePassword, $ConfigureUsers**

$PasswordEntered безопасность

Используется для ввода допустимого пароля.

Использование `$PasswordEntered`

Описание Этот тэг всегда выводится как пустая строка. Поле связи дисплея с тэгом **$PasswordEntered** всегда пустое. Поскольку этот тэг всегда возвращает пустую строку, команды изменения данных никогда его не переключают. Можно использовать для создания пользовательского окна входа в систему.

Примечание. Когда **$PasswordEntered** имеет допустимое значение, **$AccessLevel** и **$Operator** получают соответствующие предопределенные значения.

Тип данных Текстовый (только запись)

См. также **$AccessLevel, $Operator, $OperatorEntered, $ChangePassword, $ConfigureUsers**

$Second система

Содержит значение секунд текущего времени.

Использование `$Second`

Описание Значение от 0 до 59.

Тип данных Целый (только чтение)

Ïðèìåð

```
Wave=100*Sin(6*$Second);
```

$StartDdeConversations система

Начинает неинициализированный диалог DDE в среде выполнения, если меню **Специальные** не активизировано. Равносильно команде WindowViewer **Специальные/Запускать неинициализированные диалоги**.

Использование `$StartDdeConversations`

Описание Установка значения 1 начинает неинициализированный диалог DDE.

Тип данных Дискретный (чтение / запись)

$System алармы

Группа алармов по умолчанию.

Использование `$System`

Описание Если имени тэга не присвоено имя конкретной Группы Алармов, ему автоматически присваивается корневая группа по умолчанию. Все определенные пользователем Группы Алармов являются подгруппами корневой группы **$System**.

Тип данных Системная группа алармов (только чтение)

Пример

```
$System.Ack = 1;{Acknowledges All Alarms}
```

$Time система

Содержит время в миллисекундах от полуночи.

Использование `$Time`

Описание Отсутствует

Тип данных Целый (только чтение)

Пример
```
Sec_Midnight = $Time/1000;{Number of Seconds since Midnight}
```

$TimeString система

Содержит время в формате, указанном в файле WIN.INI.

Использование `$TimeString`

Описание Формат времени устанавливается в Панели управления Windows или путем двойного нажатия на поле времени в правом углу панели задач.

Тип данных Строковый (только чтение)

Пример
```
BatchStartString = $TimeString;
```

$Year система

Содержит текущий год в четырехзначном формате.

Использование `$Year`

Описание Указывает текущий год в следующем формате:

1990

Тип данных Целый (только чтение)

Пример
```
CurrentYear = $Year;
NoYrsTill2000 = 2000 - CurrentYear;
```

Г Л А В А 2

Поля тэгов

Система InTouch использует поля тэгов для доступа к свойствам объектов, таких как тэги, исторические тренды, управляющие объекты Windows. Поля тэгов существуют для того, чтобы можно было посмотреть и изменить эти свойства. В зависимости от объекта, доступ к этим свойствам может осуществляться одним из двух способов: непосредственно с помощью синтаксиса *Tagname.field* или косвенно через функции сценария.

Все объекты, за исключением управляющих объектов Windows и объектов распределенных алармов, используют синтаксис *Tagname.Field*. Чтобы получить доступ к свойствам, просто наберите имя объекта и за ним свойство, которое вас интересует в сценарии или анимационной связи. Например, чтобы разрешить изменять аварийные уставки HiHi для тэга Analog_Tag, нужно назначить анимационную связь аналогового ввода экранной кнопке и ввести Analog_Tag.HiHiLimit как выражение. Во время работы системы оператор просто нажмет на эту кнопку и введет новое значение для аварийной уставки HiHi тэга Analog_Tag.

Элементы управления окна и объекты распределенных алармов используют функцию GetPropertyX и SetPropertyX, где X - тип данных свойства (D=Discrete, I=Integer, M=Message). Эти функции также можно использовать в сценариях и анимационных связях для доступа к требуемым свойствам.

☞ Подобную информацию о функциях сценария *GetProperty* и *SetProperty* можно найти в главе 3 данного руководства.

Типы тэгов

При описании тэгов в базе данных InTouch необхоимо каждому тэгу присвоить тип в соответствии с его назначением. Если тэг должен считывать значения, исходящие из других приложений Windows или поступающие в них, например, с сервера ввода/вывода или на сервер ввода/вывода, то это должен быть тэг внешнего типа. Ниже описаны все типы тэгов InTouch и их применение.

Внутренние тэги (Memory)

Внутренние тэги (соответственно своему названию) хранятся внутри приложения InTouch. Их используют для того, чтобы создавать в системе константы и модели. С их помощью можно также создавать вычисляемые переменные, доступные из других программ Windows. Например, можно описать внутренний тэг с исходным значением 3.1416 либо сохранить рецепты в группе внутренних тэгов. При моделировании можно использовать внутренние тэги для управления фоновой работой Quick-сценария. Например, можно описать внутренний тэг "COUNT", изменяемый в Quick-сценарии действия с целью создания различных анимационных эффектов для текущей СТАДИИ (STEP) процесса. Существует 4 типа внутренних тэгов:

Memory Discrete (Внутренний дискретный)

Внутренний дискретный тэг со значением либо 0 (ложно, выключить), либо 1 (истинно, включить).

Memory Integer (Внутренний целый)

32-битное целое значение со знаком между –2,147,483,648 и 2,147,483,647.

Memory Real (Внутренний действительный)

Внутренний тэг с плавающей (десятичной) точкой. Значение плавающей точки может быть между -3.4e38 и 3.4e38. Все вычисления с плавающей точкой производятся с 64-битным разрешением, но результат хранится в 32-битном.

Memory Message (Внутренний текстовый)

Тэг с текстовой строкой, которая может содержать до 131 символа.

Внешние тэги (I/O)

Все тэги, которые считывают или записывают свои значения в другом приложении Windows, относятся к типу внешних тэгов. Сюда относятся все входы и выходы программируемых контроллеров, компьютеров управления процессами, а также данные из узлов сети. Доступ к внешним тэгам осуществляется при помощи коммуникационных протоколов Microsoft Dynamic Data Exchange (DDE) и Wonderware SuiteLink.

При изменении значения внешнего тэга чтения/записи оно немедленно записывается в удаленное приложение. Тэг можно также обновить из удаленного приложения путем изменения в этом приложении записи, с которой этот тэг связан. По умолчанию все тэги внешнего типа настроены на чтение и запись. Однако их можно ограничить только чтением путем выбора параметра Read Only в диалоговом окне **Tagname Dictionary**. Существует 4 типа внешних тэгов:

I/O Discrete (Внешний дискретный)

Дискретные внешние тэги со значением либо 0 (Ложно, Выключить), либо 1 (Истинно, Включить).

I/O Integer (Внешний целый)

32-битное целое значение со знаком между –2,147,483,648 и 2,147,483,647.

I/O Real (Внешний действительный)

Тэги с плавающей (десятичной) точкой. Значение плавающей точки может находиться в пределах +3.4e38. Все вычисления с плавающей точкой производятся с 64-битным разрешением, но результат хранится в 32-битном.

I/O Message (Внешний текстовый)

Внешний тэг с текстовой строкой, включающей не более 131 символа.

Косвенные тэги Indirect Discrete, Indirect Analog, Indirect Message

Косвенные типы тэгов позволяют создать одно окно и переназначить тэги этого окна для нескольких источников. Например, можно создать Quick-сценарий изменения данных, который изменит источник для всех тэгов в окне на основе измененного значения.

Прочие типы тэгов

Существует несколько специальных типов тэгов для выполнения сложных операций, таких как динамическое отображение алармов, создание архивных трендов, отслеживание или управление тэгами каждого пера архивного тренда. Существуют также косвенные типы тэгов, которые можно использовать для переназначения тэга на множественные источники. Эти специальные типы тэгов описаны ниже.

Group Var (Групповая переменная)

Тип **Group Var** присваивается тэгам с заданной группой алармов для создания динамического отображения аларма, протокола дисковых операций или печати. С помощью тэгов типа **Group Var** можно создать окно аларма или протокол аларма, отображающий все алармы, связанные с заданной групповой переменной. Можно сменить отображаемые или протоколируемые алармы путем назначения другой группы алармов для тэга **Group Var**.

Тэги типа **Group Var** могут также использоваться для создания кнопок, с помощью которых оператор выбирает алармы различных участков завода для отображения в одном окне аларма. В тэгах **Group Var** можно применять все **.поля**, связанные с группами алармов.

🕮 Более подробную информацию об алармах можно найти в главе 7 «*Руководства пользователя InTouch*».

Hist Trend (Архивный тренд)

Для создания архивных трендов InTouch использует тэги типа **Тренд архива**. В тэгах этого типа допускаются все **.поля**, связанные с архивными трендами.

Tag ID (Идентификатор тэга)

Это специальный тип тэгов, предназначенный для работы с объектами архивных трендов. Тэги типа **Tag ID** используются для поиска информации о тэгах, отображаемых в архивном тренде. В большинстве случаев тэги типа **Tag ID** можно использовать для отображения имени тэга, связанного с определенным пером или для выбора другого тэга для данного пера.

SuperTags (Супертэги)

Супертэги InTouch позволяют создавать составные типы тэгов. Можно определять супертэги, состоящие не более чем из 64 тэгов-членов и 2 уровней вложения. Тэги-члены работают точно так же, как обычные тэги. Они поддерживают тренды, алармы и все **.поля** тэгов.

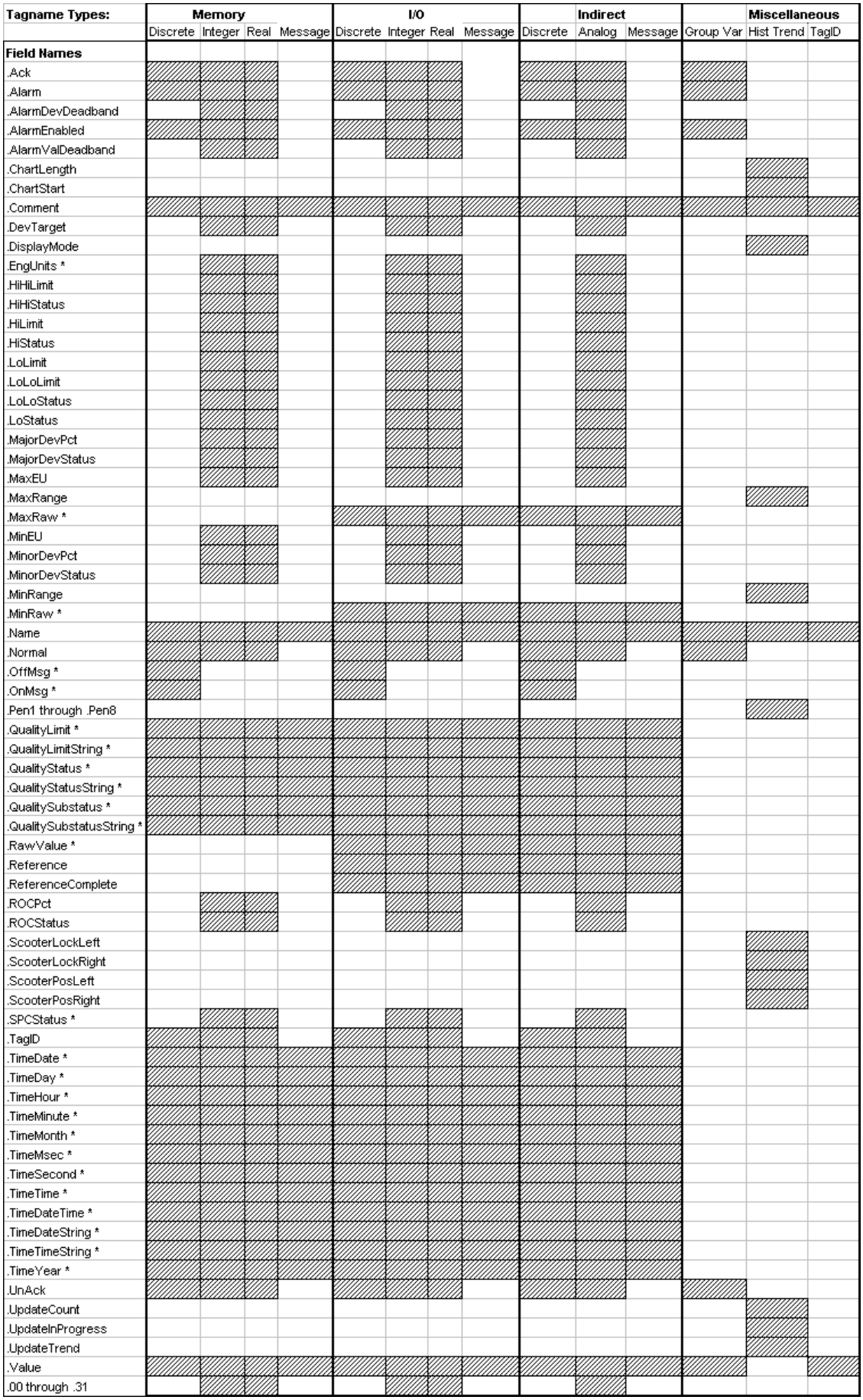
🕮 Более подробную информацию по супертэгам можно найти в «*Руководствве пользователя InTouch*».

Таблица соответствия типов тэгов и полей тэгов

Таблица на следующей странице показывает соответствие типов тэгов и полей тэгов. Приведенная ниже сноска относится к таблице на следующей странице.

* Поддерживается только в InTouch 7.0. Если клиент запрашивает эти точки памяти как элементы данных ввода/вывода, View (сервер) прикрепляет свою локальную метку времени в момент передачи значения клиенту. Однако с локальной точки зрения внутренние тэги всегда имеют ХОРОШЕЕ качество и

НЕ имеют меток времени. Это значит, что внутренние тэги НЕ обновляются в зависимости от изменения значений при помощи сценариев или связей.


Tagname Types:
Memory
I/O
Indirect
Miscellaneous
Discrete
Integer
Real
Message
Discrete
Integer
Real
Message
Discrete
Analog
Message
Group Var
Hist Trend
TagID
Field Names
.Ack
.Alarm
.AlarmDevDeadband
.AlarmEnabled
.AlarmValDeadband
.ChartLength
.ChartStart
.Comment
.DevTarget
.DisplayMode
.EngUnits *
.HiHiLimit
.HiHiStatus
.HiLimit
.HiStatus
.LoLimit
.LoLoLimit
.LoLoStatus
.LoStatus
.MajorDevPct
.MajorDevStatus
.MaxEU
.MaxRange
.MaxRaw *
.MinEU
.MinorDevPct
.MinorDevStatus
.MinRange
.MinRaw *
.Name
.Normal
.OffMsg *
.OnMsg *
.Pen1 through .Pen8
.QualityLimit *
.QualityLimitString *
.QualityStatus *
.QualityStatusString *
.QualitySubstatus *
.QualitySubstatusString *
.RawValue *
.Reference
.ReferenceComplete
.ROCPct
.ROCStatus
.ScooterLockLeft
.ScooterLockRight
.ScooterPosLeft
.ScooterPosRight
.SPCStatus *
.TagID
.TimeDate *
.TimeDay *
.TimeHour *
.TimeMinute *
.TimeMonth *
.TimeMsec *
.TimeSecond *
.TimeTime *
.TimeDateTime *
.TimeDateString *
.TimeTimeString *
.TimeYear *
.UnAck
.UpdateCount
.UpdateInProgress
.UpdateTrend
.Value
.00 through .31

.Ack алармы

Управляет состоянием квитирования локальных алармов.

Использование

```
Tagname.Ack=1
```

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный аналоговый или Переменная группы.

Комментарий

Установите это поле тэгов в значение 1, чтобы подтвердить любой возникший аларм, связанный с указанной группой алармов или указанным тэгом аларма. Если указанный тэг является Переменной группы или входит в Группу алармов, то все неквитированные алармы, связанные с тэгами внутри указанной группы, будут квитированы. Если же указан тэг любого другого типа, подтверждается только неквитированный аларм, связанный с этим тэгом. Установка поля в значение, отличное от 1, не имеет смысла, и результат будет неопределенным.

Типы данных

Дискретный (чтение / запись)

Значения

1

Примеры

Следующее выражение подтверждает аларм, связанный с тэгом "Tag1".

```
Tag1.Ack=1;
```

В следующем примере подтверждаются все алармы в группе алармов **PumpStation**:

```
PumpStation.Ack=1;
```

Для квитирования алармов можно использовать косвенную переменную AlarmGroup (с помощью GroupVar). Например:

```
StationAlarms.Name = "PumpStation";
```

Где **StationAlarms** определяется как тэг типа GroupVar и затем связывается с **PumpStation**. Следующее выражение похоже на предыдущее, но оно используется для квитирования любого неквитированного аларма из группы алармов **PumpStation**, в данный момент времени связанной с групповой переменной под названием **StationAlarms**.

```
StationAlarms.Ack=1;
```

Примечание. Для поля **.Ack** существует противоположное поле **.Unack**. При возникновении неквитированного аларма **.Unack** устанавливается на **1**. **.Unack** может затем использоваться в связях анимации или сценариях условия для переключения любых неквитированных алармов.

См. также

.Alarm и .Unack

.Alarm аларmы

Равен 1, когда возникает состояние аларма для указанного тэга.

Использование *Tagname*`.Alarm`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный аналоговый или Переменная группы.

Комментарий Обычно это неизменяемое поле тэгов равно 0. Когда существует состояние аларма для указанного тэга, система устанавливает значение 1 в это поле. Оно будет оставаться равным 1, пока существует состояние аларма . Если указанный тэг относится к групповой переменной или к Группе алармов, поле .Alarm будет установлено равным 1, если любой из тэгов внутри группы находится в состоянии аларма. Для поля **.Alarm** существует противоположное поле **.Normal**.

Типы данных Дискретный (только чтение)

Значения 0 или 1 (дискретные)

Примеры Приведенное ниже выражение даст состояние Истина, если с тэгом «Tag1» связан активный аларм:

```
IF (Tag1.Alarm == 1) THEN
```

Содержимое выражения IF-THEN будет выполняться, если внутри группы **PumpStation** будут активные алармы.

```
IF (PumpStation.Alarm == 1) THEN
   MyAlarmMessage="The pumping station currently has an
ALARM!";
ENDIF;
```

Это поле тэгов не связано с полями **.Ack** и **.UnAck**. Следовательно, если даже активный аларм был квитирован, это поле останется равным 1.

См. также **.Ack, .Normal, Ack()**

.AlarmDevDeadband алармы

Считывает и записывает размер мертвой зоны отклонений (в процентах) для алармов малого и большого отклонения.

Использование *Tagname*`.AlarmDevDeadband`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения, сделанные в этом поле с помощью анимации или сценариев в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения 0 through 100 (целые)

Пример Это выражение устанавливает процентное значение мертвой зоны отклонений равным 25%:

```
Tagname.AlarmDevDeadband=25;
```

См. также **.AlarmValDeadband**

.AlarmEnabled алармы

Разрешает и/или запрещает обслуживание локальных событий и алармов.

Использование *Tagname*`.AlarmEnabled`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Переменная группы.

Комментарий Когда **.AlarmEnabled** установлен равным 0, все события и алармы игнорируются. Они не сохраняются в буфере, не записываются на диск. Чтобы избежать потери данных, важно , как только это станет возможным, снова разрешить события/алармы.

Типы данных Дискретный (чтение / запись)

Значения 0 = Запрещает алармы и события

1 = Разрешает алармы и события (*по умолчанию*)

Примеры Это выражение запрещает все события и алармы, связанные с тэгом «Tag1».

```
Tag1.AlarmEnabled=0;
```

Содержимое выражения IF-THEN будет обрабатываться, если события и алармы разрешены для группы алармов под названием **PumpStation**.

```
IF (PumpStation.AlarmEnabled == 1) THEN
   MyAlarmMessage="The Events and Alarms for the Pump
   Station are enabled"
ENDIF;
```

См. также **.Alarm**

.AlarmValDeadband алармы

Считывает и записывает значение мертвой зоны алармов.

Использование *Tagname*`.AlarmValDeadband`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример Это выражение устанавливает значение мертвой зоны отклонений для тэга «Tag1» равным 25:

```
Tag1.AlarmValDeadband=25;
```

См. также **.AlarmDevDeadband**

.ChartLength архив

Регулирует длину (диапазон в секундах) времени на графе архивного тренда.

Использование *Tagname*.**ChartLength**

Параметры	Описание
Tagname	Любой тэг типа Тренд архива.

Комментарий Это поле для чтения/записи используется для установки или проверки значения длины графа в экране архивного тренда. Длина графа (Chart Length) выражается в секундах. Длина определяется как временной интервал, отображаемый в данный момент на графе архивного тренда. Более конкретно, для вычисления длины графа архивного тренда используется следующее:

```
ChartLength=(Штамп даты/времени на правом краю графа) -
(Штамп даты/времени на левом краю графа);
```

Поскольку штампы даты/времени выражаются в секундах, считая с полуночи 1/1/70, результат вычисляется как "количество секунд между левым и правым краями графа".

При сложении или вычитании **.ChartLength** важно помнить, что вы имеете дело с секундами. Поэтому, если нужно отнять "2 часа" от текущего значения **.ChartLength**, необходимо преобразовать "2 часа" в секунды перед вычислением, т.е.: (2 часа) * (60 минут/час) * (60 секунд/мината) = 7200 секунд.

Типы данных Целый (чтение / запись)

Значения Любое положительное целое

Пример Следующее выражение задает длину графа равной 1 часу:

```
HtTagname.ChartLength=3600 {60 минут * 60 секунд/минута};
```

Следующее выражение прокручивает граф влево на половину:

```
HtTagname.ChartStart=HtTagname.ChartStart -
HtTagname.ChartLength / 2;
```

Следующее выражение прокручивает граф влево на 10%:

```
HtTagname.ChartStart=HtTagname.ChartStart - (.10 *
HtTagname.ChartLength);
```

См. также **.ChartStart**

.ChartStart архив

Регулирует длину (диапазон в секундах) времени на графе архивного тренда.

Использование *Tagname*.ChartStart

Параметры	Описание
Tagname	Любой тэг типа Тренд архива.

Комментарий Это считываемое и записываемое поле тэгов используется для установки (или проверки) значения временного диапазона графа архивных трендов. Это значение выражается в секундах. Длина (диапазон) определяется как количество времени, выводимое на дисплей в текущий момент для графа архивного тренда. Более конкретно, для извлечения этой длины из архивного тренда применяются вычисления.

Типы данных Целый (чтение / запись)

Значения Любое положительное целое

Пример Это выражение приводит диапазон графа к 1 часу:

```
HtTagname.ChartStart=HtTagname.ChartStart + 60;
```

.Comment тэги

Содержит значение поля комментария, определяемое для каждого тэга в словаре тэгов.

Использование *Tagname*.Comment

Параметры	Описание
Tagname	Любой тэг.

Комментарий Поскольку это поле является записываемым, его содержимое влияет только на текущую (в режиме исполнения) копию комментария. Для того, чтобы сделать постоянное изменение комментария, вы должны воспользоваться WindowMaker. Это поле используется системой распределенных алармов для комментирования алармов.

Типы данных Текстовый (только чтение)

Значения Любая строка, содержащая от 1 до 50 символа. См. примечание по ограничению длины строк ниже.

Примечание. Поле комментария тэга может быть изменено только в словаре тэгов. Ввод значения в поле **.Comment** в среде WindowViewer не приведет к записи нового значения в базе данных тэгов..

Пример Следующее выражение компонует строку (содержимое тэга сообщения), собирая имя тэга и поле комментария тэга:

```
OperatorMessage=MyTag.Name + " has a comment of: " +
MyTag.Comment;
```

Примечание. Это поле записывает значение в базу данных среды выполнения. Оно не сохраняется в словаре тэгов.

.DevTarget аларmы

Это поле предназначено для чтения/записи целевых величин алармов малого и большого отклонения.

Использование *Tagname*`.DevTarget`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример Это выражение устанавливает значение DevTarget для тэга **MyTag** равным величине 500:

```
MyTag.DevTarget=500;
```

.DisplayMode архив

Определяет способ, который будет использоваться при выводе на дисплей значений тренда.

Использование *Tagname*`.DisplayMode`

Параметры	Описание
Tagname	Любой тэг типа Тренд архива.

Комментарий Отсутствует

Типы данных Аналоговый (чтение / запись)

Значения 1 = Выводит min/max для каждого периода выборки тренда (*по умолчанию*)

2 = Выводит среднее для каждого периода выборки тренда в виде графа.

3 = Выводит среднее для каждого периода выборки тренда в виде столбиковой диаграммы.

Пример Это выражение указывает, что архивный тренд, связанный с именем **MyHistTrendTag**, будет выведен в виде столбиковой диаграммы:

```
MyHistTrendTag.DisplayMode=3;
```

.EngUnits тэги

Это поле обеспечивает доступ к единицам измерения аналогового тэга, заданным в словаре тэгов.

Использование *Tagname*`.EngUnits`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это текстовое поле, не влияющее на масштаб, преобразование или формат значения тэга.

Типы данных Текстовый (чтение / запись)

Примечание. Значения, записываемые в это поле, не сохраняются.

Значения Любая строка длиной от 0 до 31 символа.

Пример

```
IF Temperature.EngUnits == "Celsius" THEN
   CALL TempConvert(Temperature);{ Можно использовать Quick-функцию для преобразования значения температуры по Цельсию в значение по Фаренгейту }
ENDIF;
```

.HiHiLimit алармы

Считывает и задает значение аварийного предела HiHiLimit.

Использование *Tagname*`.HiHiLimit`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример Это выражение увеличивает на 5 значение HiHiLimit (аварийный предел) для тэга под названием **MyTag**:

```
MyTag.HiHiLimit=MyTag.HiHiLimit + 5;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiLimit, .HiStatus, .HiHiStatus**

.HiHiStatus аларmы

Определяет, существует ли аларм HiHi - превышение второго верхнего предела.

Использование *Tagname*`.HiHiStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма, заданное пределом HiHi для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма. Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма не наступило

1 = Указанное состояние аларма наступило

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение .HiHiStatus тэга **MyTag** равно 1. Когда тэг **MyTag** превысит значение верхнего предела, тело условного оператора IF-THEN начнет выполняться:

```
IF (MyTag.HiHiStatus == 1) THEN
OperatorMessage="MyTag has gone into High-High Alarm";

ENDIF;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiLimit, .HiHiLimit, HiStatus**

.HiLimit алармы

Считывает и задает значение верхнего аварийного предела HiLimit для проверки нарушения.

Использование *Tagname*`.HiLimit`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример
```
Temperature.HiLimit = 212;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiHiLimit, HiStatus, .HiHiStatus**

.HiStatus алармы

Определяет, существует ли аларм Hi - превышение верхнего аварийного предела.

Использование *Tagname*`.HiStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма, заданное пределом Hi для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма. Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма не наступило

1 = Указанное состояние аларма наступило

Пример
```
IF (MotorAmps.HiStatus == 1) THEN
   CALL PumpShutdown( ); {Quick-функции, используемые для
очистки ввода насосного мотора}
ENDIF;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiLimit, .HiHiLimit, .HiHiStatus**

.LoLimit алармы

Считывает и задает значение нижнего аварийного предела LowLimit для проверки нарушения.

Использование *Tagname*`.LoLimit`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример Это выражение уменьшает на 10 значение LoLimit (нижнего аварийного предела) для тэга под названием **MyTag**:

```
MyTag.LoLimit=MyTag.LoLimit - 10;
```

См. также **.Alarm, .Ack, .LoLoLimit, .LoStatus, .HiLimit, .HiHiLimit, HiStatus, .HiHiStatus**

.LoLoLimit алармы

Считывает и задает значение второго нижнего аварийного предела LoLoLimit для проверки нарушения.

Использование *Tagname*`.LoLoLimit`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Аналоговый (чтение / запись)

Значения Должны быть в диапазоне, установленном для тэга.

Пример Это выражение уменьшает на 10 значение LoLoLimit (нижнего аварийного предела) для тэга под названием **MyTag**:

```
MyTag.LoLoLimit=MyTag.LoLoLimit - 10;
```

См. также **.Alarm, .Ack, .LoLimit, .HiLimit, .HiHiLimit, HiStatus, .HiHiStatus**

.LoLoStatus аларmы

Определяет, существует ли аларм LoLo – нарушение второй нижнего аварийного предела.

Использование *Tagname*`.LoLoStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма , заданное аварийным пределом LoLo для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма . Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма отсутствует

1 = Указанное состояние аларма присутствует

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение **.LoLoStatus** тэга **MyTag** равно 1. Когда тэг **MyTag** превысит значение верхнего аварийного предела, тело условного выражения IF-THEN начнет выполняться:

```
IF (MyTag.LoLoStatus == 1) THEN
OperatorMessage="MyTag has gone into Low-Low Alarm";
ENDIF;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiLimit, .HiHiLimit, HiStatus, .HiHiStatus**

.LoStatus алармы

Определяет, существует ли аларм Lo - нарушение нижнего аварийного предела.

Использование *Tagname*`.LoStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма , заданное аварийным пределом Lo для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма . Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма отсутствует

1 = Указанное состояние аларма присутствует

Пример

```
IF (WaterLevelTank1.LoStatus == 1) THEN
   PumpShutdown = 1;
   WaterFillValue = 1;
ENDIF;
```

См. также **.Alarm, .Ack, .LoLimit, .LoLoLimit, .HiLimit, .HiHiLimit, HiStatus, .HiHiStatus**

.MajorDevPct алармы

Считывает и задает процентное значение аварийного предела Major Deviation (большое отклонение).

Использование *Tagname*`.MajorDevPct`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через анимацию или сценарии в WindowViewer.

Типы данных Действительный (чтение / запись)

Значения 0 to 100%

Пример Это выражение устанавливает значение предела большого отклонения для тэга под названием **MyTag** равным 25 %:

```
MyTag.MajorDevPct=25;
```

См. также **.MajorDevStatus**

.MajorDevStatus аларmы

Определяет, существует ли состояние аларма большого отклонения (Major Deviation) для указанного тэга.

Использование *Tagname*`.MajorDevStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма отсутствует

1 = Указанное состояние аларма присутствует

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение **.MajorDevStatus** тэга **MyTag** равно 1. Когда тэг **MyTag** превысит значение предела отклонения, тело условного оператора IF-THEN начнет выполняться:

```
IF (MyTag.MajorDevStatus == 1) THEN
OperatorMessage="MyTag has gone into a Major Deviation
Alarm";
ENDIF;
```

Комментарий Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

См. также **.MajorDevPct**

.MaxEU тэги

Максимальные значения (в единицах измерения) для указанного тэга.

Использование *Tagname*`.MaxEU`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Когда в систему поступает необработанное значение тэга через DDE (Динамический обмен данными), иногда бывает разумным преобразовать это значение в единицы измерения. В качестве типичного примера допустим, что Программируемым Логическим Контроллером было считано значение измерителя уровня. Преобразователь уровня посылает сигнал в диапазоне от 4 до 20 мА. Контроллер PLC преобразует это значение (с помощью цифро-аналогового преобразователя) в целую величину от 0 до 4096. Эта величина считывается по тэгу «TankTwoLevel» («УровеньРезурвуара2»).

Вывод на дисплей такого необработанного значения (от 0 до 4096) не несет оператору никакой полезной информации. Поэтому необходимо откалибровать это значение в соответствии с диапазоном в единицах измерения. Для этого следует корректно задать поля минимального и максимального значения единиц измерения. В нашем примере, если необработанное значение 0 (4 мА) означает «0 Галлон», а значение 4096 (20 мА) означает «100 Галлон», то надо сделать следующие установки, чтобы отображать правильное значение на экране:

```
Minimum Raw=0,Maximum Raw z= 4096
```

```
Minimum Engineering Units=0,Maximum Engineering Units=100
```

При таких установках, когда придет необработанное значение 4096 с первичного датчика, на дисплее будет выдано значение 100.

Типы данных Действительный (только чтение)

Значения Зависит от указанного типа тэга.

Пример Поскольку это поле тэгов имеет атрибут «только на чтение», ему нельзя присваивать значения. Поля **.MinEU** и **.MaxEU** почень полезны для вывода на дисплей и использования их в вычислениях, так как оператору легче понять уже вычисленное значение.

```
DialogValueEntry ("IO_Point_717", IO_Point_717.MinEU,
IO_Point_717.MaxEU, "Please Enter a New Value:");
```

См. также **.MinEU, .MinRaw, .MaxRaw, .RawValue**

.MaxRange архив

Представляет процентный диапазон архивного тренда, который должен быть выведен на дисплей для каждого архивируемого тэга.

Использование *Tagname*`.MaxRange`

Параметры	Описание
Tagname	Любой тэг типа Тренд архива.

Комментарий Поскольку одновременно можно выводить на дисплей много разных тэгов, невозможно задать диапазон **.MinRange** и **.MaxRange** в единицах измерения, так как каждый тэг может иметь (и обычно имеет) совершенно различные единицы измерения. Поэтому для каждого тэга минимальное и максимальное значение диапазона выражены в процентах от физического диапазона. При таком способе, независимо от истинных единиц измерения тэга, архивный тренд просто показывает тэги в процентном выражении от реального диапазона.

Типы данных Действительный (чтение / запись)

Значения Значения **.MinRange** и **.MaxRange** лежат в пределах от 0 до 100 и выражены в процентах. Значение **.MinRange** должно быть меньше, чем **.MaxRange**. Если какому-либо из этих полей присвоено значение меньше 0 или больше 100, они урезаются до 0 или 100. Если значение **.MinRange** будет больше или равно значению **.MaxRange**, не будет выведено никаких данных.

Пример Это выражение устанавливает процент максимального значения диапазона для архивного тренда, связанного с тэгом «MyHistTrendTag», равным 25%:

```
MyHistTrendTag.MaxRange=25;
```

См. также **.ChartStart, .ChartLength, .DisplayMode, .MinEU, .MaxEU**

.MaxRaw тэги

Верхнее пороговое значение для максимального необработанного значения, полученного от сервера ввода-вывода клиентом в WindowViewer. Значение поля .MaxRaw исходит из поля Max Raw, заданного в словаре тэгов для указанного внешнего тэга. Любое необработанное значение, превышающее этот порог, обрезается до указанного значения.

Использование *tagname*`.MaxRaw`

Параметры	Описание
Tagname	Любой тэг типа Внешний Дискретный, Косвенный Дискретный, Внешний Целый, *Внутренний* Действительный, Косвенный Аналоговый, Внешний Текстовый и Косвенный Текстовый.

Комментарий Это неизменяемое поле отображает верхнее пороговое значение Max Raw.

Типы данных Действительный или Целый (только чтение)

Значения Любое аналоговое значение.

Пример Следующее выражение можно использовать для того, чтобы определять выход значения за допустимые пределы и усекать его.

```
IF ((Temp01.RawValue > Temp01.MaxRaw) OR (Temp01.RawValue <
Temp01.MinRaw))THEN
   Show "Instrument Failure Window";
ENDIF;
```

См. также **.MinRaw, .RawValue, .MinEU, .MaxEU**

.MinEU тэги

Минимальные значения (в единицах измерения) для указанного тэга.

Использование *Tagname*`.MinEU`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Когда в систему поступает необработанное значение тэга через DDE (Динамический обмен данными), иногда бывает разумным преобразовать это значение в единицы измерения. В качестве типичного примера допустим, что Программируемым Логическим Контроллером было считано значение измерителя уровня. Преобразователь уровня посылает сигнал в диапазоне от 4 до 20 мА. Контроллер PLC преобразует это значение (с помощью цифро-аналогового преобразователя) в целую величину от 0 до 4096. Эта величина считывается по тэгу «TankTwoLevel» («УровеньРезурвуара2»).

Вывод на дисплей такого необработанного значения (от 0 до 4096) не несет оператору никакой полезной информации. Поэтому необходимо откалибровать это значение в соответствии с диапазоном в единицах измерения. Для этого надо корректно задать поля минимального и максимального значения единиц измерения. В нашем примере, если необработанное значение 0 (4 мА) означает «0 Галлон», а значение 4096 (20 мА) означает «100 Галлон», то надо сделать следующие установки, чтобы отображать правильное значение на экране:

```
Minimum Raw=0,Maximum Raw = 4096
Minimum Engineering Units=0,Maximum Engineering Units=100
```

При таких установках, когда придет необработанное значение 4096 с первичного прибора, на дисплее будет выдано значение 100.

Типы данных Действительный для действительных тэгов и целый для целых тэгов (только чтение)

Значения Зависит от указанного типа тэга.

Пример

```
AbsoluteTagRange = (Tag.MaxEU - Tag.MinEU);
```

См. также **.MaxEU, .MinRaw, .MaxRaw, .RawValue**

.MinorDevPct аларmы

Считывает и задает процентное значение аварийного предела *Minor Deviation* (малого отклонения).

Использование *Tagname*`.MinorDevPct`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Установка параметров Сохранения в словаре тэгов позволит автоматически сохранять любые изменения в этом поле, сделанные через связи анимации или сценарии в WindowViewer.

Типы данных Действительный (чтение / запись)

Значения 0 -- 100%

Пример Это выражение устанавливает значение предела малого отклонения для тэга под названием **MyTag** равным 25 %:

```
MyTag.MinorDevPct=25;
```

См. также **.MinorDevStatus**

.MinorDevStatus аларmы

Определяет, существует ли состояние аларма незначительного отклонения для указанного тэга.

Использование *Tagname*`.MinorDevStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма отсутствует

1 = Указанное состояние аларма присутствует

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение .MinorDevStatus тэга **MyTag** равно 1. Когда тэг **MyTag** превысит значение предела отклонения, тело условного оператора IF-THEN начнет выполняться/.

```
IF (MyTag.MinorDevStatus == 1) THEN
OperatorMessage="MyTag has gone into a Minor Deviation
Alarm";
ENDIF;
```

Комментарий Это поле обычно используется в сочетании с полями .Alarm и .Ack для определения точной причины состояния аларма конкретного тэга в системе.

См. также **.MinorDevPct**

.MinRange архив

Представляет процентный диапазон архивного тренда, который должен быть выведен на дисплей для каждого архивируемого тэга.

Использование *Tagname*`.MinRange`

Параметры	Описание
Tagname	Любой тэг типа Тренд архива.

Комментарий Поскольку одновременно можно выводить на дисплей много разных тэгов, невозможно задать диапазон **.MinRange** и **.MaxRange** в единицах измерения, так как каждый тэг может иметь (и обычно имеет) совершенно различные единицы измерения. Поэтому для каждого тэга минимальное и максимальное значение диапазона выражены в процентах от физического диапазона. При таком способе, независимо от истинных единиц измерения тэга, архивный тренд просто показывает тэги в процентном выражении от реального диапазона.

Типы данных Действительный (чтение / запись)

Значения Значения **.MinRange** и **.MaxRange** лежат в пределах от 0 до 100 и выражены в процентах. Значение **.MinRange** должно быть меньше, чем **.MaxRange**. Если какому-либо из этих полей присвоено значение меньше 0 или больше 100, они урезаются до 0 или 100. Если значение **.MinRange** будет больше или равно значению **.MaxRange**, не будет выведено никаких данных.

Пример

```
TotalChartHeight=MyHistTrendTag.MaxRange -
MyHistTrendTag.MinRange;
```

См. также **.ChartStart, .ChartLength, .DisplayMode, .MinEU, .MaxEU, .MaxRange**

.MinRaw тэги

Нижнее пороговое значение для максимального необработанного значения, полученного от сервера I/O клиентом в WindowViewer. Значение поля .MinRaw исходит из поля Min Raw, заданного в словаре тэгов для указанного тэга I/O. Любое необработанное значение ниже этого порога обрезается до указанного значения.

Использование *tagname*`.MinRaw`

Параметры	Описание
Tagname	Любой тэг типа Внешний Дискретный, Косвенный Дискретный, Внешний Целый, Внутренний Действительный, Косвенный Аналоговый, Внешний Текстовый и Косвенный Текстовый.

Комментарий Это неизменяемое поле отображает верхнее пороговое значение Min Raw.

Типы данных Действительный или Целый (только чтение)

Значения Любое аналоговое значение.

Пример Следующее выражение можно использовать для того, чтобы определять выход значения за допустимые пределы и усекать его.

```
IF ((Temp01.RawValue > Temp01.MaxRaw) OR (Temp01.RawValue <
Temp01.MinRaw))THEN
   Show "Instrument Failure Window";
ENDIF;
```

См. также **.MaxRaw, .RawValue, .MinEU, .MaxEU**

.Name тэги

Содержит имя указанного тэга в форме строки ("String").

Использование *Tagname*`.Name`

Параметры	Описание
Tagname	Любой тип тэга.

Типы данных Текстовый (чтение / запись)

Значения Присвоить значение полю .Name можно только в тэгах косвенного типа, включающих Group Var и Tag ID. Формат присвоения значения этому полю должен соответствовать правилам имен тэгов, а именно, должны использоваться допустимые для тэгов символы, и первый символ должен быть алфивитным.

Комментарий Это поле имеет атрибут «только на чтение» для следующих типов тэгов:

Тренд архива, любой внутренний тип, любой внешний тип

Это поле имеет атрибут чтение/запись для следующих типов тэгов:

Group Var, Tag ID, любой косвенный тип

В реализации «только на чтение» это поле полезно для указания имени тэга без необходимости заключать его в кавычки, например, MyTag.Name то же самое, что **MyTag**. Когда к WindowMaker обращаются, чтобы обновить статистику использования тэгов («Update Use Counts»), и он сканирует тэги, он не может «увидеть» тэг, если он заключен в кавычки. Следовательно, если тэг используется только в кавычках, то выполнив обновление статистики использования, а затем удаление неиспользуемых тэгов («Delete unused tags»), система удалит этот тэг, так как будет считать его «неиспользуемым».

В реализации «на чтение и запись» это поле еще более полезно. Оно позволяет модифицировать идентичность указанного тэга. Когда этому полю присваивается имя другого тэга, в предположении, что оба тэга принадлежат к одному типу, указанный тэг фактически становится другим тэгом. Таким способом можно изменить тэг, на который ссылается тэг косвенного типа. Кроме того, можно изменить Группу алармов, присвоенную тэгу типа групповой переменной (Group Var).

Пример

Это выражение «только на чтение» использует имя тэга для составления сообщения оператору:

```
MyMessageTag="The tag named  " + MyTag.Name + " has a
comment of " + MyTag.Comment;
```

Следующее выражение «на чтение и запись» изменяет группу алармов, на которую ссылается тэг типа групповой переменной с именем «MyGroupVarTag»:

```
MyGroupVarTag.Name="SomeAlarmGroup";
```

Сценарий: допустим, что вы создаете графическое окно для отображения трех ключевых статистических показаний нефтяной скважины, но всего имеется 100 таких скважин. Следующий сценарий изменения данных позволит обновлять графическое окно, независимо от того, какую скважину выберет оператор. Тэгом для этого сценария будет OilWellNumber.

```
IndOilWellPump.Name = "OilWellPump" + Text(OilWellNumber,
"#");
IndOilWellTEP.Name = "OilWellTemp" + Text(OilWellNumber,
"#");
IndOilWellPressure.Name = "OilWellPressure" +
Text(OilWellNumber, "#");
```

Для выполнения этого сценария следует создать кнопку с аналоговой связью ввода по касанию, используя OilwellNumber в поле тэга. В среде выполнения сценарий будет выполнен, когда оператор нажмет эту кнопку и введет номер скважины.

.Normal алармы

.Normal равно 1, если нет алармов для указанного тэга.

Использование *Tagname*`.Normal`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Переменная группы.

Комментарий Это поле тэгов используется в сценариях или анимационных связях типа «вывод» для указания того, что заданный тэг в «нормальном» состоянии (нет активных алармов). Если становится активным один или более алармов, это поле становится равным нулю.

Типы данных Дискретный (только чтение)

Значения 0 = Один или более алармов активны для указанного тэга

1 = Нет активных алармов для указанного тэга (*по умолчанию*

Пример Блок функций «THEN» приведенного ниже выражения IF-THEN начнет выполняться в том случае, если нет алармов, связанных с тэгом под именем «Tag1». Если же для этого тэга возникнет один или более аларм, будет выполнятся тело «ELSE»:

```
IF (Tag1.Normal==1) THEN
MyOperatorMessage="Tag1 is OK - No alarms associated
with it";

ELSE
MyOperatorMessage="Tag1 has one or more alarms
active!";
ENDIF;
```

См. также **.Alarm**

.OffMsg тэги

Поле .OffMsg позволяет обращаться к сообщению "off", заданному для дискретного тэга в словаре тэгов.

Использование *Tagname*`.OffMsg`

Параметры	Описание
Tagname	Любой тэг типа Дискретный.

Типы данных Текстовый (чтение / запись)

Примечание. Значения, записываемые в это поле, не сохраняются.

Значения Любая строка длиной от 0 до 15 символов

Пример Если MyDiscrete имеет значение 0, следующее выражение присвоит строку OffMsg, заданную в словаре тэгов для MyDiscrete, тэгу StateMessage.

```
StateMessage=Dtext (MyDiscrete, MyDiscrete.OnMsg,
MyDiscrete.OffMsg);
```

См. также **.OnMsg**

.OnMsg тэги

Поле .OnMsg позволяет обращаться к сообщению "on", заданному для дискретного тэга в словаре тэгов.

Использование *Tagname*`.OnMsg`

Параметры	Описание
Tagname	Любой тэг типа Дискретный.

Типы данных Текстовый (чтение / запись)

Примечание. Значения, записываемые в это поле, не сохраняются.

Значения Любая строка длиной от 0 до 15 символов.

Пример

```
IF IndAnalog.OnMsg == "Running" THEN
   TypeOfTag = "IndAnalog was assigned a Motor Starter";
ENDIF
```

См. также **.OffMsg**

.Pen1 - .Pen8 архив

Содержит идентификатор тэга, выводимого на перо архивного тренда.

Использование

```
Tagname{.Pen1 | .Pen2 | .Pen3 | .Pen4 | .Pen5 | .Pen6 | .Pen7 | .Pen8};
```

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий

Ввиду сложности использования полей .PenX, рекомендуем использовать функции **HTSetPenName** и **HTGetPenName** там, где это возможно.

Примечание. Полю **.Pen*X*** можно присвоить только локальные тэги. Аргумент "provider.tag" использовать нельзя. "provider.tag" может использоваться только с **HTSetPenName**.

Для того, чтобы понять, как действуют эти поля тэгов, полезно поместить на экран Historical Trend Wizard и «разобрать» его на части.

Типы данных

TagID (чтение / запись)

Значения

Тип этих полей - TagID (идентификатор тэга). Это означает, что этому полю тэгов может быть присвоен идентификатор тэга. Вы не можете присвоить имя тэга этому полю. Вы должны присвоить этому полю связанное с тэгом поле .TagID. Возможно, вам полезно будет обратиться к описанию поля .TagID, чтобы лучше понять, что здесь имеется в виду. В общих словах, тэг типа TagID можно присвоить только другому тэгу типа TagID. Их нельзя смешивать с другими типами, если не добавлять расширение поля .TagID к другим тэгам.

Несмотря на то, что это поле считается разрешенным на чтение и запись, его значение не может быть непосредственно выведено на экран.

Примеры

Приводимое ниже выражение используется для присвоения нового тэга полю Pen1 архивного тренда, связанного с тэгом типа Historical Trend с именем «MyHistTrendTag». При обработке этого выражения в сценарии поле Pen1, связанное с архивным трендом с именем «MyHistTrendTag», начнет выводить на дисплей архивные данные тэга «MyLoggedTag». Обратите внимание на то, как мы должны добавить поле .TagID к имени «MyLoggedTag» для того, чтобы присвоить его полю Pen1. Это потому, что поле тэгов «Pen1» имеет тип «TagID», и, поскольку вы должны обеспечить совпадение типов при выполнении присвоения (напр., DiscreteTag = DiscretTag приемлемо, в то время как DiscreteTag = MessageTag недопустимо и фактически не имеет смысла!), этому полю тэгов надо присвоить объект типа TagID.

```
MyHistTrendTag.Pen1=MyLoggedTag.TagID;
```

А что, если мы захотим вывести на дисплей ИМЯ тэга, которое было только что присвоено «MyHistTrendTag.Pen1» ? Эта информация будет весьма полезна оператору. Кроме того, прекрасно было бы вывести имя тэга в надписи около архивного тренда для справки. Чтобы сделать это, казалось бы естественным вывести значение «MyHistTrendTag.Pen1» в поле связи MessageDisplay, однако, как можно убедиться, если вы попытаетесь так сделать, у вас ничего не выйдет. Почему ? Реальное значение поля Pen1 это целое число, предствляющее адрес в памяти, используемой программой WindowViewer; что не имеет особого смысла выводить на диспей. Чтобы двигаться дальше, нам необходимо сделать небольшое вспомогательное действие. Сначала создайте новый тэг типа TagID; назовите его «Pen01». Запишите следующее выражение под предыдущим примером:

```
Pen01=MyHistTrendTag.Pen1;
```

В первом примере мы присвоили «MyLoggedTag» полю Pen1 тэга «MyHistTrendTag». В этом примере мы продвинулись на шаг дальше, присвоив значение поля Pen1 тэга «MyHistTrendTag», которое уже равно полю TagID тэга «MyLoggedTag», тэгу Pen01. Для чего мы сделали это? Дело в том, что нам необходимо добраться до поля «.Name». Мы не может просто выдать на дисплей «MyHistTrendTag.Name» потому, что всегда будет выводиться «MyHistTrendTag». А если мы попытаемся вывести на экран «MyHistTrendTag. Pen1», редактор связи не позволит нам сделать это, поскольку Pen1 принадлежит типу TagID, который нельзя выводить на дисплей. Поэтому нам нужен вспомогательный тэг для наведения мостов над разрывом, существующим между системой дисплеев и системой архивных трендов. Присваивая значение «MyHistTrendTag. Pen1» тэгу «Pen01» типа TagID, мы, по сути, присваиваем «MyLoggedTag» тэгу «Pen01». На самом деле, мы могли бы выполнить ту же самую задачу одной строчкой сценария. Функционально они одинаковы:

```
Pen01=MyLoggedTag.TagID;
```

Тогда почему же мы выбрали первое выражение, а не второе? Потому что первое — более гибкое. Кроме того, можно просто заменить «MyHistTrendTag. Pen1» на «tagname» в сценарии изменения данных и записать команду: Pen01=MyHistTrendTag.Pen1; . В этом случае, не зная даже, какой тэг сейчас присвоен Pen1, вы сможете присвоить Pen01 тот же тэг. Это позволит вам всегда выводить на экран ИМЯ тэга, присвоенного полю Pen1 архивного тренда, даже если кто-то сделал изменения в системе либо посредством обычного интерфейса пользователя, либо с помощью сценариев.

Продвигаясь еще на один шаг в рассмотрении примера, в определенный момент вам может потребоваться вывести на экран минимальное и максимальное значения в единицах измерения для указанного тэга с архивным трендом. Мастер тренда архива делает это превосходно. Мы предлагаем, потратив небольшое время, вызвать этот мастер на экран и посмотреть, как он был спроектирован. Это позволит вам понять, как работают эти поля тэгов.

Поскольку тэги типа TagID, такие как «Pen01», не имеют единиц измерения и остальных полей тэгов, обычно имеющихся у тэга, мы не можем их использовать для вывода единиц измерения связанного с ними тэга. Например, мы не можем вывести «Pen01.MaxEU», так как редактор связи посетует на то, что .MaxEU не является допустимым полем для Pen01, поскольку он типа TagID.

Чтобы обойти это затруднение, нам просто нужен вспомогательный тэг для осуществления преобразования между типом TagID и системой дисплеев. Сначала создайте тэг типа Косвенный Аналоговый и назовите его «IndirectAnalogPen1». В том же сценарии по изменению данных, где записан пример #2, запишите следующее выражение:

```
IndirectAnalogPen1.Name=Pen01.Name;
```

Имя Pen01 (в данный момент в нашем примере его значение равно «MyLoggedTag») присвоено косвенному тэгу «IndirectAnalogPen1». Теперь мы можем иметь доступ ко всем полям тэгов, обычно присутствующим у аналоговых тэгов. Так что теперь мы можем вывести на экран с помощью анимационной связи типа «Вывод аналогового выражения» следующее:

```
IndirectAnalogPen1.MaxEU
```

Это выведет на экран максимальные единицы измерения тэга, связанного в данный момент с тэгом «IndirectAnalogPen1». Поскольку мы выше присвоили

ему значение Pen01.Name в сценарии, мы можем быть уверенными в том, что будут выведены единицы измерения тэга, присвоенного в текущий момент полю Pen1 «MyHistTrendTag».

См. также **.TagID, HTGetPenName( ), HTSetPenName( )**

.Quality тэги

Для более четкого представления о поле качества (Quality) приведем краткое определение стандарта качества, используемого фирмой Wonderware. Стандарт качества данных Wonderware основан на нормативах качества OLE для управления процессами (OPC), которые, в свою очередь, базируются на спецификациях Fieldbus Data Quality.

Флаги качества отражают состояние качества определенных значений элементов данных. Это позволяет разработчикам легко определять требуемый уровень функциональности при проектировании как серверных и клиентских приложений.

Младшие 8 бит байта у флагов качества определены в форме трех битовых полей; качество (Quality), подстатус (Substatus) и предельный статус (Limit status):

QQSSSSLL

Поле Quality позволяет использовать значение качества внешнего тэга, получаемого от сервера ввода/вывода.

Примечание. При разрыве соединения с сервером ввода-вывода поля качества сбрасываются на изначальные нулевые значения. Флаг **.ReferenceComplete** при этом также сбрасывается на ноль.

Использование *Tagname*`.Quality`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 255

Пример

```
IF I0Tag.Quality <> 192 THEN
   LogMessage("This data is not Good!");
ENDIF;
```

Серверы ввода/вывода Wonderware могут предоставлять по запросу приложений-клиентов следующие шесть (6) взаимоисключающих качественных состояний данных:

	Hex-значение .QualiyLimit	**Биты качества OPC** MSByte XXXXXXXX	**Биты качества OPC** LSByte QQSSSSLL	**Биты качества OPC** .QualityStatus	.QualitySubStatus	
1. Good	0x00C0	00000000	00000000	Q=3	S=0	L=0
2. Clamped High	0x0056	00000000	00000000	Q=1	S=5	L=2
3. Clamped Low	0x0055	00000000	00000000	Q=1	S=5	L=1
4. Cannot Convert	0x0040	00000000	00000000	Q=1	S=0	L=0
5. Cannot Access Point	0x0004	00000000	00000000	Q=0	S=1	L=0
6. Communications Failed	0x0018	00000000	00000000	Q=0	S=6	L=0

Если приложение-клиент не имеет связи с сервером, .QualityStatus равно 0.

	0x0000	00000000	00000000	Q=0	S=0	L=0

Каждое из этих качественных состояний может предоставляться при следующих условиях:

1. Good

- Соединение с сервером исправно.
- Контроллер PLC успешно принял запрос и вернул правильный пакет ответа.
- Если был передан запрос на запись, запись была выполнена успешно.
- Данные ответного пакета были преобразованы без ошибок.

Пример В ответ на запрос регистра, содержащего десятичное 10, возвращено значение 0x0000A).

2. Clamped High

- Соединение с сервером исправно.
- Контроллер PLC успешно принял запрос и вернул правильный пакет ответа.
- Регистр был считан или изменен без ошибок.
- Полученное значение было усечено до заданного предела, поскольку оно было слишком высоким.
- В случае текстовой строки, она была обрезана по длине.

Пример Беззнаковое 16-битное целое усекается до 65535.

3. Clamped Low

- Соединение с сервером исправно.
- Контроллер PLC успешно принял запрос и вернул правильный пакет ответа.
- Регистр был считан или изменен без ошибок.
- Полученное значение было усечено до заданного предела, поскольку оно было слишком низким.

Пример Беззнаковое 16-битное целое усекается до 0.

4. Cannot Convert

- Соединение с сервером исправно.
- Контроллер PLC успешно принял запрос и вернул правильный пакет ответа.
- Данные PLC не удалось преобразовать в требуемый формат.
- Неудачное преобразование может произойти в числе других по следующим причинам:
  - Сервер возвращает константу вместо данных или только информацию о качестве.
  - Данные оказываются непригодными.
  - Не известно, являются ли значения слишком высокими или слишком низкими.
  - Данные, полученные от PLC, имеют неверный тип данных.
  - Возвращается число с плавающей точкой, не являющееся значением (например: Not A Number).

Пример Регистр BCD в PLC возвращает значение 0x000A.

5. Cannot Access Point

- Контроллер PLC успешно принял запрос и вернул правильный пакет ответа.
- PLC сообщает, что не может получить доступ к запрошенной точке.
- Невозможность доступа может быть вызвана в числе других следующими причинами:
  - Запрошенный элемент данных отсутствует в памяти PLC.
  - Элемент недоступен в данный момент (например, заблокирован ввиду перегрузки ресурсов).
  - Элемент имеет неверный формат или тип данных.
  - Сделана попытка записи в элемент, предназначенный только для чтения.
  - Как правило, ошибка в одном элементу приводит оказывает влияние сразу на несколько других элементов – поскольку серверы используют технологию блочного опроса. Например, если ошибка произошла в одном элементе блока, состоящего из 10, то PLC признает весь блок непригодным. Сервер сообщит о недопустимом качестве всех элементов блока.
  - Данные непригодны.

Пример Делается попытка чтения из R40001, но R40001 не определен в адресном пространстве памяти PLC.

6. Communications Failed

- Любое сочетание следующих причин:
  - Связь с данными потеряна.
  - Тема (Topic) находится в режиме медленного опроса (или эквивалентном).
  - Не получено сообщений подтверждения связи.
  - Нехватка ресурсов на сервере. Например, не достаточно памяти для резидентной программы (или драйвера).
  - Нехватка ресурсов на линии связи.
  - Линия связи не готова.
  - Все каналы связи заняты.
  - Сеть не может маршрутизировать сообщение в PLC

Пример Делается попытка передать сообщение на выключенный PLC.

См. также **.QualityLimit, .QualityStatus, .QualitySubstatus**

.QualityLimit тэги

Целочисленное поле, отображающее предел качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualityLimit`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения (LL)

0	Not Limited
1	Low Limited
2	High Limited
3	Constant

См. также **.Quality**

.QualityLimitString тэги

Отображает строку предельного качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualityLimitString`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Текстовый (только чтение)

См. также **.QualityLimit, .Quality**

.QualityStatus тэги

Целочисленное поле, отображающее статус качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualityStatus`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Комментарий Битовое поле подстатуса (SSSS) зависит от значения поля качества (QQSSSSLL).

Типы данных Целый (только чтение)

Значения (SSSS)

0	Плохое
1	Неопределенное
3	Хорошее

См. также **.QualitySubStatus, .Quality**

.QualityStatusString тэги

Отображает строку статуса качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualityStatusString`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Текстовый (только чтение)

См. также **.QualityStatus, .QualitySubStatus, .Quality**

.QualitySubstatus тэги

Целочисленный тэг, отображающий подстатус качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualitySubstatus`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения (SSSS) и (QQ)

Подстатус (SSSS) ПЛОХОГО качества (QQ=0):

0	Общий статус
1	Ошибка конфигурации
2	Нет соединения
3	Отказ устройства
4	Отказ датчика
5	Последнее известное значение
6	Отказ связи
7	Нет готовности

Подстатус (SSSS) НЕОПРЕДЕЛЕННОГО качества (QQ=1):

0	Общий статус
1	Последнее использованное значение
4	Неточные показания датчика
5	Превышение инженерных единиц
6	Поднормальное

Подстатус (SSSS) ХОРОШЕГО качества (QQ=2):

0	Общий статус
6	Локальный принудительный

См. также **.QualityStatus, .Quality**

.QualitySubstatusString тэги

Отображает строку подстатуса качества значения ввода/вывода, переданного сервером ввода/вывода при исправном соединении.

Использование *Tagname*`.QualitySubstatusString`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Текстовый (только чтение)

См. также **.QualityStatus, .QualitySubstatus, .Quality**

.RawValue тэги

Фактическое значение, полученное от сервера ввода /вывода клиентом WindowViewer. Поле необработанного значения позволяет обращаться к значению тэга внешнего типа до того, как InTouch произведет масштабирование.

Использование `tagname.RawValue`

Параметры	Описание
Tagname	Любой тэг типа Внешний Дискретный, Косвенный Дискретный, Внешний Целый, *Внутренний* Действительный, Косвенный Аналоговый, Внешний Текстовый или Косвенный Текстовый.

Комментарий Это поле, предназначенное только для чтения, отображает фактическое дискретное или аналоговое значение внешнего тэга до того, как InTouch произведет масштабирование.

Типы данных Данные любого типа. Например, действительный для действительных тэгов, дискретный для дискретных тэгов и т.д. (только чтение)

Значения Дискретные или аналоговые — необходимо указывать.

Пример Следующее выражение позволяет определить, находится ли тэг за пределами нормального рабочего состояния.

```
IF ((IOtag.RawValue > IOtag.MaxRaw) OR (IOTag.RawValue <
IOTag.MinRaw))THEN
   AlarmMessage = "Sensor is out of calibration or requires
      replacement.";
ENDIF;
```

См. также **.MinRaw, .MaxRaw, .MinEU, .MaxEU**

.Reference тэги

Позволяет оператору динамически изменять в режиме выполнения имя доступа DDE (Access Name) и/или элемент доступа (Item).

Использование *TagName*`.Reference`

Параметры	Описание
Tagname	Любой тэг типа Внешний Дискретный, Косвенный Дискретный, Внешний Целый, Внешний Действительный, Косвенный Аналоговый, Внешний Текстовый или Косвенный Текстовый.

Комментарий Это поле тэгов предоставляет удобный способ динамически изменять имя доступа DDE (Access Name) или элемента доступа (Item) указанного тэга.

Типы данных Текстовый (чтение / запись)

Значения Любая строка, содержащая имя доступа и/или элемент.

Пример Следующее выражение присваивает полю элемента внешнего целого тэга имя, сгенерированное текстовой функцией и определенное значением внутреннего целого тэга. Оно будет использовано в сценарии условия, использующем **.ReferenceComplete** для цикличного прохода по серии тэгов:

```
MyIOTag.Reference="Accessname,"+ "R" + Text( “#” MyIndex, );
{ Если MyIndex=40001, в результате ITEM будет:  R40001 }
```

.ReferenceComplete тэги

Возвращает подтверждение, если запрашиваемый элемент такой же, как и значение, отраженное в поле **.Value**.

Использование *TagName*`.ReferenceComplete;`

Параметры	Описание
Tagname	Любой тэг типа Внешний Дискретный, Косвенный Дискретный, Внешний Целый, Внешний Действительный, Косвенный Аналоговый, Внешний Текстовый или Косвенный Текстовый.

Комментарий Это поле тэгов полезно при использовании поля **.Reference**. Поле **.ReferenceComplete** показывает, что элемент или имя доступа изменились и, что более важно, было получено новое значение из нового источника данных. Даже если значение нового источника данных совпадает со старым источником данных, когда обновление происходит из нового источника данных, это поле устанавливается равным 1.

Во время фактического процесса обновления, со времени модификации .Reference до времени первого обновления от нового источника данных, значение этого поля будет установлено системой равным 0.

Типы данных Дискретный (только чтение)

.ROCPct алармы

Считывает и записывает скорость изменения для проверки аларма.

Использование *Tagname*.**ROCPct**

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Значение этого поля тэгов выражено в процентах. Это поле прямо соответствует аналогичному полю, сконфигурированному в разделе алармов в словаре тэгов. Пользователь может выводить на дисплей это поле или использовать его в сценариях. Кроме того, это значение можно изменить в режиме исполнения, чтобы учесть изменения, произошедшие в технологическом процессе.

Типы данных Целый (чтение / запись)

Значения 0 - 100%

Пример Это выражение устанавливает значение скорости изменения в процентах для тэга под названием **MyTag** равным 25%:

```
MyTag.ROCPct=25;
```

См. также **.ROCStatus**

.ROCStatus аларmы

Определяет, существует ли состояние аларма из-за превышения скорости изменения (Rate-of-Change) для указанного тэга.

Использование *Tagname*.**ROCStatus**

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма.

Типы данных Дискретный (только чтение)

Значения 0 = Указанное состояние аларма отсутствует

1 = Указанное состояние аларма присутствует

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение **.ROCStatus** тэга **MyTag** равно 1. Когда тэг **MyTag** войдет в состояние аларма скорости изменения, тело условного оператора IF-THEN начнет выполняться:

```
IF (MyTag.ROCStatus == 1) THEN
OperatorMessage="MyTag has gone into a Rate-Of-Change-
Alarm";

ENDIF;
```

Комментарий Это поле обычно используется в сочетании с полями **.Alarm** и **.Ack** для определения точной причины состояния аларма конкретного тэга в системе.

См. также **.ROCPct**

.ScooterLockLeft архив

Установка значения этого поля равным 1 (Истина) не разрешит правому визиру двигаться влево от позиции левого визира (не позволит правому визиру обойти левый).

Использование *Tagname*`.ScooterLockLeft`

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий В общем случае было бы полезно не давать пользователю возможности передвигать правый визир левее текущей позиции левого визира. Если левый визир не заблокирован, правый визир выровняет позиции левого и правого визира, когда догонит левый.

Типы данных Дискретный (чтение / запись)

Значения 0 = FALSE (Ложь) = Правый визир может перекрывать позицию левого

1 = TRUE (Истина) = Правый визир не может перекрывать позицию левого

Пример Это выражение установит правый визир, связанный с архивным трендом по имени «MyHistTrendTag», таким образом, что он не сможет двигаться влево от текущей позиции левого визира.

```
MyHistTrendTag.ScooterLockLeft=1;
```

См. также **.ScooterPosRight, .ScooterPosLeft, .ScooterLockRight**

.ScooterLockRight архив

Установка значения этого поля равным 1 (Истина) не разрешит левому визиру двигаться вправо от позиции правого визира (не позволит левому визиру обойти правый).

Использование *Tagname*`.ScooterLockRight`

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий В общем случае было бы полезно не давать пользователю возможности передвигать левый визир правее текущей позиции правого визира. Если правый визир не заблокирован, левый визир догонит правый и поравняется с ним.

Типы данных Дискретный (чтение / запись)

Значения 0 = FALSE (Ложь) = Левый визир может перекрывать позицию правого

1 = TRUE (Истина) = Левый визир не может перекрывать позицию правого

Пример Это выражение установит левый визир, связанный с архивным трендом по имени «MyHistTrendTag», таким образом, что он не сможет двигаться вправо от текущей позиции правого визира.

```
MyHistTrendTag.ScooterLockRight=1;
```

См. также **.ScooterPosRight, .ScooterPosLeft, .ScooterLockLeft**

.ScooterPosLeft архив

Считывает и записывает позицию левого визира.

Использование *Tagname*.**ScooterPosLeft**

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий Это поле с атрибутом «на чтение и запись» динамически управляет позицией левого визира. Пользователь может использовать эту функцию в сценариях, чтобы получить текущую позицию левого визира или присвоить значение этому полю для настройки позиции левого визира на другую точку тренда.

Это поле чаще всего используется в сочетании с семейством функций **HTGetValue()**. Этим функциям надо указать, какой архивный тренд запрашивается, а также текущую позицию визиров тренда. Смотрите приведенные ниже примеры.

Типы данных Действительный (чтение / запись)

Значения От 0.0 до 1.0; где 0 - левая граница графа архивного тренда, а 1.0 – его правая граница.

Пример Это выражение назначает новое положение левому визиру. Левый визир передвинется на позицию 34% общей длины графа от крайней левой стороны графа архивного тренда, связанного с архивным трендом по имени «MyHistTrendTag»:

```
MyHistTrendTag.ScooterPosLeft=.34;
```

В следующем выражении функция **HTGetValueAtScooter()** используется для извлечения значения Pen1 в текущей позиции левого визира. Поскольку изменение любой переменной внутри списка тэгов функции вызывает переоценку функции, каждый раз, когда позиция левого визира будет меняться, значение этого выражения будет меняться.

```
MyRealTag=HTGetValueAtScooter( MyHistTrendTag,
   MyHistTrendTag.UpdateCount,1,
   MyHistTrendTag.ScooterPosLeft,1,"PenValue");
```

См. также **.ScooterPosRight, .ScooterLockLeft, ScooterLockRight**

.ScooterPosRight архив

Считывает и записывает позицию правого визира.

Использование *Tagname*`.ScooterPosRight`

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий Это поле с атрибутом «на чтение и запись» динамически управляет позицией правого визира. Пользователь может использовать эту функцию в сценариях, чтобы получить текущую позицию правого визира или присвоить значение этому полю для настройки позиции правого визира на другую точку тренда.

Это поле чаще всего используется в сочетании с семейством функций **HTGetValue()**. Этим функциям надо указать, какой архивный тренд запрашивается, а также текущую позицию визиров тренда. Смотрите приведенные ниже примеры.

Типы данных Действительный (чтение / запись)

Значения От 0.0 до 1.0; где 0 - правая сторона графа архивного тренда, а 1.0 -левая сторона графа архивного тренда.

Примеры Это выражение назначает новое положение правому визиру. Правый визир передвинется на позицию 34% общей длины графа от крайней правой стороны графа архивного тренда, связанного с архивным трендом по имени «MyHistTrendTag»:

```
MyHistTrendTag.ScooterPosRight=.34;
```

В следующем выражении функция **HTGetValueAtScooter()** используется для извлечения значения Pen1 в текущей позиции правого визира. Поскольку изменение любой переменной внутри списка тэгов функции вызывает переоценку функции, каждый раз, когда позиция правого визира будет меняться, значение этого выражения будет меняться.

```
MyRealTag=HTGetValueAtScooter( MyHistTrendTag,
   MyHistTrendTag.UpdateCount,2,
   MyHistTrendTag.ScooterPosRight,1,"PenValue");
```

См. также **.ScooterPosLeft, .ScooterLockLeft, ScooterLockRight**

.SPCStatus SPC

Определяет, существует ли состояние аларма SPC для указанного тэга.

Примечание. Это поле аларма применимо только, если установлена программа SPC (Статистическое управление процессом).

Использование *Tagname*`.SPCStatus`

Параметры	Описание
Tagname	Любой тэг типа Целый, Действительный или Косвенный Аналоговый.

Типы данных Дискретный (только чтение)

Значения 0 = Состояние аларма SPC отсутствует

1 = Имеет место состояние аларма SPC

Комментарий Это неизменяемое поле тэгов обычно равно 0. Когда наступает состояние аларма для указанного тэга, система устанавливает значение этого поля в 1. Это поле будет содержать единицу до тех пор, пока не исчезнет состояние аларма.

Пример Содержимое выражения IF-THEN будет обрабатываться только, если значение **.SPCStatus** («Statistical Process Control Alarm» - аларм статистического управления процессом) тэга **MyTag** равно 1. Когда тэг **MyTag** войдет в состояние аларма SPC, тело условного оператора IF-THEN начнет выполняться.

```
IF (MyTag.SPCStatus == 1) THEN
OperatorMessage="MyTag has gone into an SPC-Alarm";
ENDIF;
```

.TagID тэги

Используется совместно с полями **.Pen1-.Pen8** архивного тренда для управления тэгами, которые выводятся на перо архивного тренда.

Использование *Tagname*`.TagID`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Дискретный или Косвенный Аналоговый.

Комментарий .TagID предоставляет указатель тэга и используется, главным образом, для присвоения тэгов перьям архивного тренда.

Типы данных TagID (только чтение)

Пример

```
MyHistTrendTag.Pen6=SomeAnalogTag.TagID;
```

См. также **.Pen1-.Pen8**

.TimeDate тэги

Целочисленное поле, отображающее количество полных дней, прошедших с момента передачи значения сервером ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeDate**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

.TimeDateString тэги

Строка, отображающая дату в формате, заданном в файле WIN.INI.

Использование *Tagname*.**TimeDateString**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Текстовый (только чтение)

.TimeDateTime тэги

Действительное поле, отображающее неполное количество дней, прошедших с момента передачи значения сервером ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeDateTime**

Параметры	Описание
Tagname	Любой тэг Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Действительный (только чтение)

.TimeDay тэги

Целочисленное поле, отображающее день, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeDay**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 1 до 31.

.TimeHour тэги

Целочисленное поле, отображающее час дня, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeHour**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 23.

.TimeMinute тэги

Целочисленное поле, отображающее минуту, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeMinute**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 59.

.TimeMonth тэги

Целочисленное поле, отображающее месяц, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeMonth**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 1 до 12.

.TimeMsec тэги

Целочисленное поле, отображающее время в миллисекундах, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeMsec**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 999.

.TimeSecond тэги

Целочисленное поле, отображающее время в секундах, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeSecond**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 59.

.TimeTime тэги

Целочисленное поле, отображающее время в миллисекундах (с полуночи), когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeTime**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения От 0 до 86399999.

.TimeTimeString тэги

Текстовое поле, отображающее время и день, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeTimeString**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Текстовый (только чтение)

.TimeYear тэги

Целочисленное поле, отображающее год в четырехзначном формате, когда значение ввода/вывода было получено от сервера ввода/вывода при исправном соединении.

Использование *Tagname*.**TimeTime**

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный, Косвенный Аналоговый или Текстовый.

Типы данных Целый (только чтение)

Значения Любой год в формате ####. Например 1998.

.Unack алармы

Считывает и записывает статус квитирования локальных алармов.

Использование

`Tagname.Unack=1`

Параметры	Описание
Tagname	Любой тэг типа Дискретный, Целый, Действительный Косвенный или Переменная группы.

Комментарий

Установите это поле тэгов в значение 1, чтобы снять статус квитирования любого аларма, связанного с указанной группой алармов или указанным тэгом аларма. Если указанный тэг является типом Переменная группы или входит в Группу алармов, то будет снято квитирование всех алармов, связанных с тэгами внутри указанной группы. Если же указан тэг любого другого типа, то квитирование снимается только с аларма, связанного с этим тэгом. Установка поля в значение, отличное от 1, не имеет смысла, и результат будет неопределенным.

Типы данных

Дискретный (чтение / запись)

Значения

1

Примеры

Следующее выражение снимает квитирование аларма, связанного с тэгом "Tag1".

```
Tag1.Unack=1;
```

В следующем примере снимается квитирование всех алармов в группе алармов **PumpStation:**

```
PumpStation.Unack = 1;
```

Это выражение похоже на предыдущее за тем исключением, что оно снимает квитирование со всех квитированных алармов группы, связанной с тэгом группы переменных "StationAlarms".

Примечание. Для поля **.Unack** существует противоположное поле **.Ack**. При возникновении квитированного аларма поле **.Ack** устанавливается на **1**.

См. также

.Ack и .Alarm

.UpdateCount архив

Увеличивается каждый раз, когда происходит обновление связанного с этим полем архивного тренда.

Использование *Tagname*.**UpdateCount**

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий Это поле тэгов связано непосредственно с подсистемой считывания архивных трендов. Как только у заданного архивного тренда запрашиваются новые данные, подсистема считывания архивных данных должна обратиться к диску и считать указанные данные. Как только процесс считывания завершится, это поле тэгов увеличит свое значение на 1. Поле **.UpdateCount** используется во многих функциях, имеющих отношение к архивным трендам, как способ инициализации пересчета значений в этих функциях.

Типы данных Целый (только чтение)

Значения Любое положительное целое

Пример Это выражение использует функцию **HTGetValueAtScooter()** для считывания значения Pen1 в текущей позиции правого визира. Поскольку изменения любой переменной в списке тэгов функции вызывает пересчет самой функции, каждый раз, когда завершается считывание (обновление) данных и увеличивается значение **.UpdateCount**, это выражение будет вычислено снова.

```
MyRealTag=HTGetValueAtScooter( MyHistTrendTag,
   MyHistTrendTag.UpdateCount,2,
   MyHistTrendTag.ScooterPosRight,1,"PenValue");
```

См. также **.UpdateInProgress, .UpdateTrend**

.UpdateInProgress архив

Равен 1, если идет процесс обновления архивного тренда; в противном случае равно 0.

Использование *Tagname*.**UpdateInProgress**

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий Это поле тэгов связано непосредственно с подсистемой считывания архивных трендов. Как только у заданного архивного тренда запрашиваются новые данные, подсистема считывания архивных данных должна обратиться к диску и считать указанные данные. Во время процесса считывания это поле тэгов равно 1. Как только процесс завершится, значение поля **.UpdateInProgress** сбросится в 0. Поле **.UpdateInProgress** используется во многих функциях, имеющих отношение к архивным трендам.

Большинство экранов архивных трендов имеют механизм, с помощью которого оператор может «прокручивать» данные тренда. По мере того, как оператор манипулирует управляющими объектами на экране, архивная подсистема должна гарантировать появление на экране текущих данных. Если оператор пролистывает тренд в том интервале времени, который не выведен на дисплей в данный момент или не находится в резидентной памяти, подсистема обратится к диску и считает требуемые данные. Поскольку это может занять некоторое время, система предоставляет способ проектировщику экрана архивных трендов дать знать оператору о том, что запрошенные данные считываются в данный момент. Без такого рода обратной связи оператор не может быть уверенным, что система выполняет затребованное действие.

Типы данных Дискретный (только чтение)

Value Values 0 = Не идет процесс обновления

1 = Идет процесс обновления

Примеры Это выражение используется для формирования выражения на надписи текстового объекта около или на кнопке прокручивания архивного тренда. Когда архивная подсистема считывает запрошенные данные, результатом этого выражения будет значение 1. В противном случае, если тренд не требует обновления, значение этого выражения будет 0.

```
MyHistTrendTag.UpdateInProgress
```

См. также **.UpdateCount, .UpdateTrend**

.UpdateTrend архив

Заставляет граф архивного тренда обновиться, используя все текущие значения.

Использование *Tagname*`.UpdateTrend`

Параметры	Описание
Tagname	Любой тэг типа *Тренд архива*.

Комментарий Архивные тренды не обновляют сами себя автоматически. Нужно изменить значение либо поля ChartStart, либо ChartLength, чтобы граф был обновлен и вывелись текущие значения для указанных тэгов. Использование поля .UpdateTrend при задании реакции на нажатие кнопки позволит оператору обновить граф, когда бы это не потребовалось в режиме исполнения.

Это можно также использовать при описании других полей тэгов, связанных с архивным трендом, которые надо изменить. При этом будет обеспечено, что на дисплей графа архивных трендов будут выводиться самые последние данные.

Установка этого поля в значение, отличное от 1 не имеет никакого смысла, и результат в этом случае не определен.

Типы данных Дискретный (только запись)

Значения 1

Пример Это выражение вызывает обновление архивного тренда «MyHistTrendTag» текущими значениями для всех тэгов:

```
MyHistTrendTag.UpdateTrend=1;
```

.Value тэги

Содержит текущее значение указанного тэга. Является также полем тэгов для каждого тэга InTouch. Если не указано любое другое поле, подразумевается поле **.Value**.

Использование *Tagname*`.Value`

Параметры	Описание
Tagname	Любой тэг кроме типа *Тренд архива*.

Комментарий Это поле является полем по умолчанию для каждого тэга InTouch. Если не указано никакое другое поле, подразумевается это. Вам редко понадобится использовать это поле, но в некоторых случаях его явное указание делает более понятным вычисления или применение тэга, и поэтому может быть полезным как средство документирования.

Типы данных Зависит от типа (такой же как) указанного тэга (на чтение и запись).

Пример Это выражение устанавливает значение внутренней переменной целого типа с именем **MyTag** равным 100.

```
Tagname.Value=100;
```

Что функционально идентично следующему:

```
Tagname=100;
```

.AlarmGroup распределенные алармы

Содержит текущий список опроса распределенных алармов.

Использование

```
[ErrorNumber=]GetPropertyM("ObjectName.AlarmGroup",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., *AlmObj_1*.
Tagname	Текстовый тэг, в котором будет содержаться значение свойства в режиме выполнения функции.

Комментарий Это неизменяемое поле тэгов содержит результат текущего опроса алармов, который используется для указанного дисплея распределенных алармов. Результатом опроса может быть список групп алармов или прямые ссылки на алармы.

Типы данных Текстовый (только чтение)

Пример Это выражение возвращает результат опроса текущих алармов, которые выводятся в объекте «AlmObj_1», в сообщение с тэгом «CurrentQuery»:

```
GetPropertyM("AlmObj_1.AlarmGroup",CurrentQuery)
```

См. также **GetPropertyM()**

.NextPage распределенные алармы

Пролистывает экран алармов на одну страницу (при полном экране алармов) вниз, когда это свойство переходит от 1 к 0.

Использование

```
[ErrorNumber=]GetPropertyD("ObjectName.NextPage",Tagname);
[ErrorNumber=]SetPropertyD("ObjectName.NextPage",Value);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., *AlmObj1*.
Tagname	Тэг дискретного типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.
Value	Дискретное значение. Или дискретный тэг InTouch, содержащий значение, которое должно быть записано в режиме выполнения функции.

Комментарий Когда это значение переходит от 1 к 0, на экран алармов будет выведена следующая страница. Когда появится следующая страница, переменная автоматически будет установлена в 1, если не достигнуто начало списка. В этом случае переменная останется равной 0.

Типы данных Дискретный (чтение / запись)

См. также **GetPropertyD(), SetPropertyD(), .PrevPage**

.NumAlarms распределенные алармы

Содержит количество алармов внутри объекта аларма.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.NumAlarms",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов «только на чтение» содержит текущее число алармов, зарегистрированных для экранов распределенных алармов с заданным именем. В это число входят не только выводимые на экран алармы, но все зарегистрированные алармы.

Типы данных Целый (только чтение)

Пример Следующее выражение возвращает текущее число алармов для объекта дисплея «AlmObj_1» в переменную целого типа под именем «AlarmCount»:

```
GetPropertyI("AlmObj_1.NumAlarms",AlarmCount);
```

См. также **GetPropertyI()**

.PageNum распределенные алармы

Содержит номер текущей страницы, выводимой в объект аларма.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.PageNum",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов «только на чтение» содержит номер выводимой в данный момент страницы на указанный экран распределенных алармов.

Типы данных Целый (только чтение)

Пример Следующее выражение возвращает текущий номер страницы алармов для объекта дисплея «AlmObj_1» в переменную целого типа под именем «AlarmPage»:

```
GetPropertyI("AlmObj_1.PageNum",AlarmPage);
```

См. также **GetPropertyI()**

.PrevPage распределенные алармы

Пролистывает экран алармов на одну страницу (при полном экране алармов) вверх, когда это свойство переходит от 1 к 0.

Использование

```
[ErrorNumber=]GetPropertyD("ObjectName.PrevPage",Tagname);
[ErrorNumber=]SetPropertyD("ObjectName.PrevPage",Value);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Дискретный тэг, в котором будет содержаться запрашиваемое свойство при выполнении функции
Value	Дискретное значение. Или дискретный тэг InTouch, содержащий значение, которое должно быть записано в режиме исполнения функции.

Комментарий Когда это значение переходит от 1 к 0, на экран алармов будет выведена предыдущая страница. Когда появится следующая страница, переменная автоматически будет установлена в 1, если не достигнуто начало списка. В этом случае переменная останется равной 0.

Типы данных Дискретный (чтение / запись)

См. также **GetPropertyD(), SetPropertyD(), .NextPage**

.PriFrom распределенные алармы

Содержит нижнее значение приоритета при опросе алармов.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.PriFrom",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит минимальное значение приоритета, используемого в указанном экране распределенных алармов при опросе алармов.

Типы данных Целый (только чтение)

Пример используемого объектом аларма под названием «AlmObj_1» и помещает результат в тэг целого типа «MinPri»:

```
GetPropertyI("AlmObj_1.PriFrom",MinPri);
```

См. также **GetPropertyI(), .PriTo**

.PriTo распределенные алармы

Содержит текущее значение верхнего фильтра приоритета запроса

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.PriTo",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит максимальное значение приоритета, используемого в указанном дисплее распределенных алармов при опросе алармов.

Типы данных Целый (только чтение)

Пример Это выражение возвращает максимальное значение приоритета запроса, используемого объектом аларма под названием «AlmObj_1» и помещает результат в тэг целого типа «MaxPri»:

```
GetPropertyI("AlmObj_1.PriTo",MaxPri);
```

См. также **GetPropertyI(), .PriFrom**

.ProviderReq распределенные алармы

Содержит количество источников аларма, требуемое при текущем опросе.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.ProviderReq",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит количество источников аларма, требуемое при текущем опросе, который используется указанным экраном распределенных алармов.

Типы данных Целый (только чтение)

Пример Это выражение возвращает количество источников аларма, требуемое при текущем опросе, используемого объектом аларма под названием «AlmObj_1». Результат записывается в тэг целого типа «TotalProv»:

```
GetPropertyI("AlmObj_1.ProviderReq",TotalProv);
```

См. также **GetPropertyI(), .ProviderRet**

.ProviderRet распределенные алармы

Содержит количество источников аларма, которые успешно возвратили результаты опроса.

Использование `[ErrorNumber=]GetPropertyI("ObjectName.ProviderRet",Tagname);`

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит количество источников аларма, которые успешно возвратили результаты опроса, используемого указанным экраном распределенных алармов.

Типы данных Целый (только чтение)

Пример Это выражение возвращает количество источников аларма, которые успешно возвратили результаты опроса, используемого объектом дисплея под названием «AlmObj_1». Результат записывается в тэг целого типа «RetProv»:

```
GetPropertyI("AlmObj_1.ProviderRet",RetProv);
```

См. также **GetPropertyI(), .ProviderReq**

.QueryState распределенные алармы

Содержит текущий фильтр опроса состояния алармов.

Использование `[ErrorNumber=]GetPropertyI("ObjectName.QueryState",Tagname);`

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только чтение» содержит текущий фильтр опроса, который используется указанным экраном распределенных алармов.

Типы данных Целый (только чтение)

Значения 0 = Все

1= Неквитированные

2 = Квитированные

Пример Это выражение получает текущий фильтр опроса для объекта дисплея под названием «AlmObj_1». Результат записывается в тэг целого типа «AlmQueryState»:

```
GetPropertyI("AlmObj_1.QueryState",AlmQueryState);
```

См. также **GetPropertyI()**

.QueryType распределенные алармы

Содержит тип текущего опроса алармов.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.QueryType",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит текущий тип опроса, который используется указанным экраном распределенных алармов.

Типы данных Целый (только чтение)

Значения 1 = История (History)

2 = Обзор (Summary)

Пример Это выражение возвращает тип текущего опроса, используемого объектом аларма под названием «AlmObj_1». Результат записывается в тэг целого типа «AlmQueryType»:

```
GetPropertyI("AlmObj_1.QueryType",AlmQueryType);
```

См. также **GetPropertyI()**

.Successful распределенные алармы

Указывает, был ли успешным результат текущего опроса.

Использование

```
[ErrorNumber=]GetPropertyD("ObjectName.Successful",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Дискретный тэг, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит состояние последнего опроса, который используется указанным экраном распределенных алармов.

Типы данных Дискретный (только чтение)

Значения 0 = Ошибка при опросе

1 = Опрос прошел успешно

Пример Это выражение возвращает состояние последнего опроса, который используется объектом аларма под названием «AlmObj_1». Результат записывается в тэг дискретного типа «AlmFlag»:

```
GetPropertyD("AlmObj_1.Successful",AlmFlag);
```

См. также **GetPropertyD()**

.TotalPages распределенные алармы

Содержит общее число страниц (при полном экране алармов) в объекте алармов.

Использование

```
[ErrorNumber=]GetPropertyI("ObjectName.TotalPages",Tagname);
```

Параметры	Описание
ObjectName	Имя объекта аларма, напр., AlmObj_1.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это поле тэгов с атрибутом «только на чтение» содержит общее число страниц алармов, содержащихся в указанном экране распределенных алармов.

Типы данных Целый (только чтение)

Пример Это выражение возвращает общее число страниц алармов, содержащихся на экране алармов «AlmObj_1», в целый тэг «AlmTotalPages»:

```
GetPropertyI("AlmObj_1.TotalPages",AlmTotalPages);
```

См. также **GetPropertyI()**

.Caption элементы управления окна

Определяет, какое «сообщение» будет выведено на дисплей в заголовке управляющего объекта "флажок" (Checkbox).

Использование

```
[ErrorNumber=]GetPropertyM ("ControlName.Caption", Tagname );
[ErrorNumber=]SetPropertyM ("ControlName.Caption",
"Message");
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., ChkBox_4.
Tagname	Тэг текстового типа, в котором будет содержаться запрошенное свойство.
"Message"	Текстовая строка в кавычках.

Комментарий Это свойство считывается / записывается как в режиме разработки, так и в режиме исполнения.

Типы данных Текстовый (чтение / запись)

Назначение Используется для флажков (Check boxes).

Пример Это выражение присваивает заголовку объекта «CheckBox_1» значение «Blue Paint Option».

```
SetPropertyM( "CheckBox_1.Caption","Blue Paint Option" );
```

См. также **GetPropertyM(), SetPropertyM()**

.Enabled элементы управления окна

Определяет, будет ли элемент управления реагировать на события, сгенерированные пользователем.

Использование

```
[ErrorNumber=] GetPropertyD("ControlName.Enabled", Tagname);
[ErrorNumber=] SetPropertyD("ControlName.Enabled", Discrete);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., ChkBox_4.
Tagname	Дискретный тэг, в котором будет содержаться запрошенное свойство.
Discrete	Дискретное значение: 0 = Управление запрещено 1 = Управление разрешено -Or- Дискретный тэг, в котором будет содержаться значение для записи в запрошенное свойство.

Комментарий Это свойство считывается / записывается как в режиме разработки, так и в режиме исполнения.

Типы данных Дискретный (чтение / запись)

Назначение Поля ввода (Text boxes), списки (List boxes), поля ввода со списком (Combo boxes), флажки (Check boxes) и переключатели (Radio buttons).

Пример Это выражение запрещает управление объекту List box под названием «ListBox_1»:

```
SetPropertyD( "ListBox_1.Enabled",0 );
```

См. также **GetPropertyD(), SetPropertyD()**

.ListCount элементы управления окна

Определяет число пунктов в списке List box или Combo box.

Использование

```
[ErrorNumber=]GetPropertyI("ControlName.ListCount",Tagname);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *ListBox_4*.
Tagname	Тэг, содержащий целое число - количество пунктов в списке.

Комментарий Это свойство доступно только в режиме исполнения.

Типы данных Целый (только чтение)

Назначение **Простые и комбинированные списки (**List boxes и Combo boxes).

Пример Это выражение извлекает количество пунктов в списке под названием «ListBox_1» и помещает значение во внутреннюю переменную целого типа под названием «MyListBoxCount».

```
GetPropertyI( "ListBox_1.ListCount",MyListBoxCount );
```

См. также **GetPropertyI()**

.ListIndex элементы управления окна

Определяет индекс (тэг или число) выбранного пункта в списке. Индекс - это порядковый номер, определяющий конкретный пункт в списке. Для списка "Combo box" индекс -1 означает, что пользователь ввел новый текст в текстовое поле ввода элемента управления.

Использование

```
[ErrorNumber=]GetPropertyI("ControlName.ListIndex",Tagname);
[ErrorNumber=]SetPropertyI("ControlName.ListIndex",Number);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *ListBox_4*.
Tagname	Тэг, содержащий целое число - количество пунктов в списке.
Number	Число, определяющее конкретный пункт в списке.

Комментарий Это свойство доступно только в режиме исполнения.

Типы данных Целый (чтение / запись)

Назначение Простые и комбинированные списки (List boxes и Combo boxes).

Пример Это выражение получает индекс выбранного пункта в списке под названием «ListBox_1» и помещает значение во внутреннюю переменную целого типа под названием «MyListBoxIndex».

```
GetPropertyI( "ListBox_1.ListIndex",MyListBoxIndex );
```

См. также **GetPropertyI(), SetPropertyI()**

.NewIndex элементы управления окна

Возвращает соответствующий целый индекс (Tagname) последнего пункта, который был добавлен в список List box или Combo box с помощью функций wcAddItem() или wcInsertItem().

Использование

```
[ErrorNumber=]GetPropertyI("ControlName.NewIndex",Tagname);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *ListBox_4*.
Tagname	Тэг, содержащий целый индекс последнего пункта, добавленного в список List box или Combo box. При пустых списках возвращается -1.

Комментарий Это свойство доступно только в режиме исполнения.

Типы данных Целый (только чтение)

Назначение Простые и комбинированные списки (List boxes и Combo boxes).

Пример Это выражение возвращает индекс последнего добавленного пункта в список с именем «ListBox_1» и помещает это значение во внутреннюю переменную целого типа с именем «NewItemIndex»:

```
GetPropertyI("ListBox_1.NewIndex",NewItemIndex);
```

См. также **GetPropertyI(), wcAddItem(), wcInsertItem()**

.ReadOnly элементы управления окна

Определяет атрибут содержимого объекта Text box - «только чтение» или «чтение и запись».

Использование

```
[ErrorNumber=]GetPropertyD("ControlName.ReadOnly", Tagname);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *Textbox_1*.
Tagname	Дискретный тэг, в котором будет содержаться запрашиваемое свойство при выполнении функции.

Комментарий Это свойство считывается / записывается как в режиме разработки, так и в режиме исполнения.

Типы данных Дискретный (только чтение)

Значения 0 = Содержимое текстового окна можно считывать и записывать

1= Содержимое текстового окна можно только считывать

Назначение **Поля ввода (**Text boxes).

Пример Это выражение устанавливает атрибут «только на чтение» текстовому окну под названием «TextBox_1»:

```
GetPropertyD( "TextBox_1.ReadOnly",A_Tagname );
```

См. также **GetPropertyD(), SetPropertyD()**

.TopIndex элементы управления окна

Определяет целый индекс самого верхнего пункта в списке.

Использование

```
[ErrorNumber=]GetPropertyI("ControlName.TopIndex", Tagname);
[ErrorNumber=]SetPropertyI("ControlName.TopIndex", Number);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *Listbox_1*.
Tagname	Тэг целого типа, в котором будет содержаться запрашиваемое свойство при выполнении функции.
Number	Номер самого верхнего пункта в списке.

Комментарий Это свойство доступно только в режиме исполнения.

Типы данных Целый (чтение / запись)

Назначение Списки (List boxes).

Пример Это выражение устанавливает верхнему пункту списка под названием «ListBox_1» значение 14:

```
SetPropertyI( "ListBox_1.TopIndex",14 );
```

См. также **GetPropertyI(), SetPropertyI()**

.Value элементы управления окна

Содержит текущее значение указанного тэга. Является также полем тэгов для каждого тэга в InTouch. Если не указано любое другое поле, подразумевается поле **.Value**.

Использование

```
[ErrorNumber=]GetPropertyM("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyM("ControlName[.Value]", Value);
[ErrorNumber=]GetPropertyI("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyI("ControlName[.Value]", Value);
[ErrorNumber=]GetPropertyD("ControlName[.Value]", Tagname);
[ErrorNumber=]SetPropertyD("ControlName[.Value]", Value);
```

Примечание. Первоначальное значение тэгов, связанных с "List box" или "Combo box" не может использоваться для инициализации значений этих элементов.

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., ChkBox_4.
[**.Value**]	Это свойство не является обязательным. Если оно не задано, функция всегда использует текущее свойство **.Value**.
Tagname	Тэг (того же типа, что и возвращаемый), содержащий значение свойства при выполнении функции.
Value	Фактическое записываемое значение или тэг (такого же типа, что и записываемое свойство), содержащий значение свойства при выполнении функции.

Комментарий

Это свойство считывается и записывается в режиме разработки и в режиме исполнения. Если доступ к полю **.Value** осуществляется путем связывания тэга со списком типа List Box или Combo Box, то это поле только на чтение. Если же поле **.Value** присвоено объектам Check box, Radio button или Text box, тогда это поле и на чтение и на запись. Значение, указанное при разработке служит значением по умолчанию в режиме исполнения.

Типы данных

Тип Текстовый (чтение/запись) для элементов Text boxes, List boxes и Combo boxes.

Тип Целый (чтение/запись) для элементов типа Radio buttons.

Тип Дискретный (чтение/запись) для элементов типа Check boxes.

Назначение

Text box, List box, Combo box, Check box, Radio button.

Пример

Это выражение устанавливает значение поля **.Value** объекта RadioButton с именем «RadioButton_1» равным 4:

```
SetPropertyI( "RadioButton_1.Value",4 );
```

См. также

GetPropertyM(), SetPropertyM(), GetPropertyI(), SetPropertyI(), GetPropertyD(), SetPropertyD()

.Visible элементы управления окна

Определяет, будет ли элемент управления видимым в окне.

Использование

```
[ErrorNumber=]GetPropertyD("ControlName.Visible",Tagname);
[ErrorNumber=]SetPropertyD("ControlName.Visible",Number);
```

Параметры	Описание
ControlName	Имя элемента управления окнами, напр., *Listbox_1*.
Tagname	Тэг (того же типа, что и возвращаемый), содержащий значение свойства при выполнении функции.
Number	Дискретное значение. Или дискретный тэг, содержащий значение свойства при выполнении функции.

Комментарий Это свойство считывается / записывается как в режиме разработки, так и в режиме исполнения.

Типы данных Дискретный (чтение / запись)

Значения 0 = Элемент управления невидим

1= Элемент управления видим

Назначение Text box, List box, Combo box, Check box, Radio button.

Пример Это выражение делает объект "Text box" с именем «TextBox_1» невидимым:

```
SetPropertyD( "TextBox_1.Visible",0 );
```

См. также **GetPropertyD(), SetPropertyD()**

Г Л А В А 3

Функции Quick-сценариев InTouch

Quick-сценарии InTouch позволяют выполнять команды и логические операции при возникновении определенных событий или условий. Например, нажимается клавиша, открывается окно, изменяется значение и тому подобное.

Quick-функции — это созданные вами сценарии, которые можно вызывать из других сценариев или выражений связей анимации. Повторяемый код хранится в одном сценарии и в одном месте, позволяя тем самым обновлять все экземпляры сценария за один сеанс редактирования.

С помощью сценариев можно создать самые разнообразные нестандартные и автоматические функции.

Abs() математическая

Возвращает абсолютное значение (беззнаковый эквивалент) заданного числа.

Синтаксис

```
Result=Abs(Number);
```

Параметры	Описание
Number	Любое число или тэг InTouch действительного или целого типа.

Комментарий Вычисляется абсолютное значение Number и возвращается в переменную Result.

Примеры

```
Abs(14) возвращает 14
Abs(-7.5) возвращает 7.5
```

Ack() алармы

Подтверждает любые неподтвержденные алармы InTouch.

Синтаксис

```
Ack Tagname;
```

Параметры	Описание
Tagname	Любой тэг InTouch , Группа алармов или Переменная группы.

Комментарий Эту функцию можно применить к тэгу типа Alarm Group (группа алармов) или Group Variable (переменная группы). Переменная группы — это тэг, имеющий имя связанной с ним группы алармов.

Примеры Эти выражения можно связать с нажимаемой кнопкой, чтобы подтверждать любые неподтвержденные алармы.

```
Ack $System; (Все алармы)
Ack Tagname; (Подтверждать алармы тэга)
Ack GroupName; (Подтверждать группу алармов)
Ack GroupVariable; (Подтверждать алармы, связанные с тэгом типа Group Variable)
```

ActivateApp() системная

Активизирует другое выполняемое в данный момент приложение Windows.

Синтаксис

```
ActivateApp TaskName;
```

Параметры	Описание
TaskName	Задача, которую активизирует эта функция.

Комментарий *TaskName* — в точности та текстовая строка, включая пробелы, которая появляется на панели задач или в окне Диспетчера задач Windows (нажать правой кнопкой мыши на панель задач Windows NT и выбрать Диспетчер задач или нажать комбинацию Ctrl+Alt+Delete).

Пример Следующее выражение проверяет наличие выполняемой командной подсказки. Если она выполняется, она переключается из фонового режима в основной и ставится в фокус. В противном случае, командная подсказка

запускается с выполнением DOS-программы edit.com в окне командной подсказки.

```
IF InfoAppActive( InfoAppTitle("cmd")) == 1 THEN
   ActiveApp InfoAppTitle("cmd");
ELSE
   StartApp "cmd /c edit";
ENDIF;
```

См. также **StartApp(), InfoAppTitle()**

almAckAll() распределенные алармы

Подтверждает все алармы в текущем опросе, включая те, что не показаны в данный момент на экране алармов.

Синтаксис `[Result=]almAckAll(ObjectName,Comment);`

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, *AlmObj_1*.
Comment	Комментарий к подтверждению аларма.

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

```
MessageTag = «Acknowledge All by» + $Operator;
AlmAckAll(«AlmObj_1», MessageTag);
```

almAckDisplay() распределенные алармы

Подтверждает самые последние возникшие алармы.

Синтаксис `[Result=]almAckDisplay(ObjectName,Comment);`

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, AlmObj_1.
Comment	Комментарий к подтверждению аралма.

☞ Описание возвращаемых сообщений об ошибках дано в Приложении А.

Пример

```
almAckDisplay(«AlmObj_1», «Display Acknowledgement»);
```

almAckRecent() распределенные алармы

Подтверждает только те алармы, которые выбраны (отмечены) на экране алармов.

Синтаксис `[Result=]almAckRecent(ObjectName,Comment);`

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, AlmObj_1.
Comment	Комментарий к подтверждению аралма.

Комментарий Информация о кодах ошибок содержится в Приложении А.

Пример

```
almAckRecent(«AlmObj_1», $DateString);
```

almAckSelect() распределенные алармы

Подтверждает только те алармы, которые выбраны (отмечены) на экране алармов.

Синтаксис

[Result=]**almAckSelect**(*ObjectName,Comment*);

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, AlmObj_1.
Comment	Комментарий к подтверждению аралма.

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

```
IF ($Hour > 0 and $Hour < 8) THEN
   AckTag = «NightShift»;
ELSE
   AckTag = «Day Shift»;
ENDIF;
AlmAckSelect («AlmObj_1»,AckTag);
```

almDefQuery() распределенные алармы

Выполняет запрос на обновление экрана алармов, используя свойства по умолчанию.

Синтаксис

[Result=]**almDefQuery**(*ObjectName*);

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, AlmObj_1.

Комментарий

Свойства запросов по умолчанию задаются на этапе разработки.

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

```
almDefQuery(«AlmObj_1»);
```

almMoveWindow()распределенные алармы

Пролистывает окно экрана алармов.

Синтаксис

```
[Result=]almMoveWindow(ObjectName,Options,Repeat);
```

Параметры	Описание
ObjectName	Заголовок объекта аларма, например, AlmObj_1.
Option	Тип выполняемого подтверждения аларма:

Type	Описание
LineDn	На одну строку вниз.
LineUp	На одну строку вверх.
PageDn	На одну страницу вниз.
PageUp	На одну страницу вверх.
Top	В самый верх списка.
Bottom	В самый низ списка.
PageRt	На одну страницу вправо.
PageLf	На одну страницу влево.
Right	К правому краю списка.
Left	К левому краю списка.

Repeat	Число повторов данной операции.

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

```
almMoveWindow(«AlmObj_1», »Bottom», 0);
almMoveWindow(«AlmObj_1», »LineDn», 3);
almMoveWindow(«AlmObj_1», »PageUp», 0);
```

almQuery() распределенные алармы

Выполняет запрос на обновление экрана алармов.

Синтаксис

```
[Result=]almQuery(ObjectName,AlarmList,FromPri,
   ToPri,State,Type);
```

Параметр	Описание
ObjectName	Заголовок объекта аларма, например, *AlmObj_1*.
AlarmList	Задает группу алармов или список групп для опроса, например, «MyGroup» или тэг типа "message".
FromPri	Начальный приоритет выводимых алармов, напр., 100 или тэг целого типа.
ToPri	Конечный приоритет выводимых алармов, напр., 900 или тэг целого типа.
State	Задает тип выводимых алармов, напр. «UnAck» (неподтвержденные) или собщение. Допустимые значения: All (все), UnAck (неподтвержденные), Ack (подтвержденные).
Type	Задает тип опроса, напр., «Hist» (архив алармов) или «Summ» (обзор алармов).

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

Это выражение отбирает все исторические алармы, указанные в группе «MyGroup» между приоритетом 500 и 600. Эти алармы будут выведены на экран алармов «AlmObj_1»:

```
AlmQuery("AlmObj_1","MyGroup",500,600,"All","Hist");
```

almSelectAll() распределенные алармы

Включает/выключает выбор всех пунктов на экране алармов.

Синтаксис

```
[Result=]almSelectAll(ObjectName);
```

Параметр	Описание
ObjectName	Заголовок объекта аларма, например, *AlmObj_1*

.

☞ Информация о кодах ошибок содержится в Приложении А.

Пример

```
If $AccessLevel > 8000 THEN
   almSelectAll(«AlmObj_1»);
   almAckSelect(«AlmObj_1», «Ack Selected by a Manager»);
ENDIF;
```

almSelectItem() распределенные алармы

Включает/выключает выбор пункта, выделенного на экране алармов.

Синтаксис

```
[Result=]almSelectItem(ObjectName);
```

Параметр	Описание
ObjectName	Заголовок объекта аларма, например, *AlmObj_1*.

☞ Информация о кодах ошибок содержится в Приложении A.

Пример

```
almSelectItem(«AlmObj_1»);
```

almShowStats() распределенные алармы

Выводит окно статистики для экрана алармов.

Синтаксис

```
[Result=]almShowStats(ObjectName);
```

Параметр	Описание
ObjectName	Заголовок объекта аларма, например, *AlmObj_1*.

☞ Информация о кодах ошибок содержится в Приложении A.

Пример

```
almSlowStats(«AlmObj_1»);
```

ArcCos() математическая

Для заданного числа от -1 до 1 (включительно) эта функция возвратит угол между 0 и 180 градусов, чей *косинус* равен заданному числу.

Синтаксис

```
Result=ArcCos(Number);
```

Параметр	Описание
Number	Любое число или тэг InTouch действительного или целого типа.

Комментарий Вычисляется значение арксинуса числа *Number*, и результат возвращается в переменную *Result*.

Пример

```
ArcCos(1) возвращает 0
ArcCos(-1) возвращает 180
```

ArcSin() математическая

Для заданного числа от -1 до 1 (включительно) эта функция возвратит угол между -90 и 90 градусов, чей *синус* равен заданному числу.

Синтаксис

```
Result=ArcSin(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Вычисляется абсолютное значение *Number* и возвращается в *Result*.

Пример

```
ArcSin(1) возвращает 90
ArcSin(-1) возвращает -90
```

ArcTan() математическая

Для заданного числа эта функция возвратит угол между -90 и 90 градусов, чей *тангенс* равен заданному числу.

Синтаксис

```
Result=ArcTan(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Вычисляется абсолютное значение *Number* и возвращается в *Result*.

Пример

```
ArcTan(1) возвращает 45
ArcTan(0) возвращает 0
```

ChangePassword() доступ

Выдает на экран диалоговое окно **Change Password** (изменить пароль), давая возможность зарегистрированному пользователю изменить свой пароль.

Синтаксис

```
[Result=]ChangePassword();
```

Параметр	Описание
[Result]	Возвращает одно из следующих целых значений: 0=Нажата кнопка Отмена. 1=Нажата кнопка OK.

Комментарий При работе приложений с сенсорным экраном (touch screen) предоставляется возможность использования алфавитно-цифровой клавиатуры.

Пример

```
Errmgs=ChangePassword();
```

Если эта функция связана с экранной кнопкой или вызывается из сценария, то она откроет диалоговое окно (с дополнительной клавиатурой), дающее пользователю приглашение ввести текущий пароль, новый пароль и подтверждение нового пароля.

Cos() математическая

Возвращает *косинус* угла, заданного в градусах.

Синтаксис

```
Result=Cos(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Вычисляется абсолютное значение *Number* и возвращается в *Result*.

Пример

```
Cos(90) возвращает 0
Cos(0) возвращает 1
Wave = 50 * Cos(6 * $Minute);
```

DialogStringEntry() прочие

Выдает на экран алфавитно-цифровую клавиатуру, давая пользователю возможность изменить текущее значение тэга типа сообщения.

Синтаксис

```
[Result=]DialogStringEntry(MessageTag_Text,UserPrompt_Text);
```

Параметр	Описание
Result	Возвращает одно из следующих целых значений: 0=Нажата кнопка Отмена. 1=Нажата кнопка OK. -1=Внутренняя ошибка. -2=Не выполнена инициализация. -3=Не определен тэг. -4=Тэг не относится к типу "message". -5=Не выполнена запись.
MessageTag_Text	Задает имя тэга типа "message", которое надо изменить. (Эта функция требует строку для тэга, поэтому вы должны указать тэг по имени в кавычках или использовать поле **.Name** без кавычек.) Допустимо также задавать указатель на тэг. См. приведенные примеры.
UserPrompt_Text	Задает сообщение пользователя сверху клавиатуры.

Комментарий

Эта функция используется преимущественно в приложениях для сенсорного экрана (touch screen).

Примеры

```
Errmsg=DialogStringEntry(MyMessageTag.Name,
"Введите новую строку...");
```

```
Errmsg=DialogStringEntry("MyMessageTag","Введите новую
строку...");
```

Параметры могут также выступать в качестве "указателей" на другие тэги, делая функцию динамической в режиме исполнения.

```
Errmsg=DialogStringEntry(MessageTagX, MessageDisplay);
```

Используя вышеприведенный пример, анимационные связи ввода могут быть присвоены каждому тэгу-указателю, позволяя пользователю модифицировать любой или все тэги функции перед выполнением функции.

Например, следующие команды вызовут на экран алфавитно-цифровую клавиатуру, предоставляя возможность модифицировать сообщение, выдавая сообщение «Введите новую стороку...» над изображением клавиатуры:

```
MessageTagX="MyMessageTag"; {присвоить строку MyMessageTag
(имя тэга, подлежащего изменению) внутреннему текстовому тэгу
MessageTagX}
```

```
MessageDisplay="Введите новую строку..."; {присвоить новую
строку внутреннему текстовому тэгу MessageDisplay}
```

```
Errmsg=DialogStringEntry(MessageTagX, MessageDisplay);
{кавычки не требуются, поскольку MessageTagX определен как
тэг текстового типа }
```

DialogValueEntry() прочие

Выводит на экран цифровую клавиатуру, предоставляя пользователю возможность изменить текущее значение дискретного, целого или действительного тэга.

Синтаксис

```
[Result=]DialogValueEntry(ValueTag_Text,
   LowLimit,HighLimit,UserPrompt_Text);
```

Параметр	Описание
Result	Возвращает одно из следующих целых значений: 0=Нажата кнопка Отмена. 1=Нажата кнопка OK. -1=Highlimit<=Lowlimit. -2=Не выполнена инициализация. -3=Тэг не определен. -4=Тэг не является дискретным, целым или действительным. -5=Не удалось произвести запись.
ValueTag_Text	Задает имя дискретного, целого или действительного тэга, который надо изменить. (Функция требует строку для этого тэга; поэтому значение надо задавать в кавычках или использовать поле **.Name** без кавычек или задать указатель на сообщение. См. примеры.)
LowLimit	Задает минимально допустимое значение для тэга. (Оно должно быть >= значениям Min Value, Min Raw, Min EU, определенным для тэга).
HighLimit	Задает максимально допустимое значение для тэга. (Оно должно быть <= значениям Max Value, Max Raw, Max EU, определенным для тэга).
UserPrompt_Text	Задает сообщение пользователя, которое будет выведено сверху клавиатуры.

Комментарий

Эта функция используется преимущественно в приложениях для сенсорного экрана (touch screen).

Примеры

```
Errmsg=DialogValueEntry(MyIntegerTag.Name,
MyIntegerTag.MinEU, MyIntegerTag.MaxEU, "Введите новое
значение...");
```

```
Errmsg=DialogValueEntry("MyIntegerTag", -100, 100, "Введите
новое значение...");
```

Параметры могут также выступать в качестве "указателей" на другие тэги, делая функцию динамической в режиме исполнения.

```
Errmsg=DialogValueEntry(TagnameX, Min, Max, MessageDisplay);
```

Используя вышеприведенный пример, можно присвоить анимационные связи ввода каждому тэгу-указателю, позволяя пользователю модифицировать любой или все тэги функции перед выполнением функции.

Например, следующие команды вызовут на экран алфавитно-цифровую клавиатуру, предоставляя возможность модифицировать тэг MyIntegerTag с ограничениями Min и Max, равными соответственно -100 и 100, выдавая сообщение «Введите новое значение...» сверху клавиатуры:

```
TagnameX="MyIntegerTag"; {присвоить строку MyIntegerTag (имя тэга, подлежащего изменению) внутреннему текстовому тэгу TagnameX}

Min=-100; {присвоить минимально допустимое значение внутреннему действительному/целому тэгу Min}

Max=100; {присвоить минимально допустимое значение внутреннему действительному/целому тэгу Max}

MessageDisplay="Введите новое значение..."; {присвоить новую строку внутреннему текстовому тэгу MessageDisplay}

Errmsg=DialogValueEntry(TagnameX, Min, Max, MessageDisplay);
{кавычки не требуются, поскольку TagnameX определен как тэг текстового типа. Путем присвоения дискретного, целого или действительного тэга для TagnameX функция изменит присвоенный тэг}
```

DText() строка

Динамически изменяет значение тэга типа сообщение, основываясь на значении дискретного тэга.

Синтаксис

```
MsgTag=DText(Discrete_Tag,OnMsg,OffMsg);
```

Параметр	Описание
MsgTag	Тэг текстового типа.
Discrete_Tag	Тэг дискретного типа.
OnMsg	Сообщение, выводимое, когда *Discrete_Tag* равен 1 (True, On, Yes).
OffMsg	Сообщение, выводимое, когда *Discrete_Tag* равен 0 (False, Off, No).

Пример

```
MessageTag=Dtext(DiscreteTag, DiscreteTag.OnMsg, DiscreteTag.OffMsg);

If Dtext(D_Tag, «True», «False») == «True» THEN
   Message=»D_Tag contains a value of 1»;
ELSE
   Message=»D_Tag contains a value of 0»;
ENDIF;
```

Exp() математическая

Возвращает значение *e*, возведенное в степень.

Синтаксис

```
Result=Exp(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Вычисляется экспонента числа *Number* и результат возвращается в переменную *Result*.

Пример

```
Exp(1) возвращает 2.718...
```

Диапазон этой функции от -88.72 до 88.72.

FileCopy() системная

Копирует исходный файл (*SourceFile)* в файл назначениея (*DestFile)* аналогично команде Copy в Dos или функции Copy в Диспетчере файлов Windows.

Синтаксис

```
FileCopy(SourceFile,DestFile,DoneTag);
```

Параметр	Описание
SourceFile	Имя исходного файла (включая полный путь).
DestFile	Имя файла назначения (включая полный путь) или имя каталога (см. приведенные примеры).
DoneTag	Имя тэга, которое используется функцией **FileCopy()** для *отчета* о прохождении процедуры копирования. Этот тэг должен быть строкой, задающей имя тэга (а не сам тэг). Таким образом, если ваш тэг называется Monitor, вы должны указать «Monitor» или *Monitor.Name*, а не просто Monitor.

Комментарий При использовании функции **FileCopy()**, она сразу же возвращает 1, если процедура копирования успешно начала выполняться. Она возвращает 0, если в данный момент выполняется другая процедура (нельзя начать новую процедуру) или -1, если возникла ошибка. Используя возвращаемое значение, можно проследить за инициализацией функции **FileCopy()**:

```
Status=FileCopy("C:\*.TXT", "C:\BACKUP", "Monitor");
```

Status - это целый тэг, куда записывается 1, -1 или 0. Функция **FileCopy()**, выполняется в фоновом режиме, поэтому она не будет мешать выполнению InTouch. Тэг DoneTag позволяет приложению или пользователю отслеживать *процесс* выполнения копирования. Таким образом можно сразу привлечь внимание пользователя, если ПОСЛЕ инициализации возникли какие-либо ошибки. Это отличается от значения функции, которое возвращается в переменную *Status*, и дает результат *инициализации* функции.

Значение *DoneTag* присваивается только после успешной инициализации функции. Это значение равно 0, пока процедура находится в стадии выполнения. Оно устанавливается равным 1, если процедура выполнилась успешно, или -1, если до завершения процедуры возникла ошибка.

Обычно, *SourceFile* и *DestFile* — это имена файлов. Однако, если копируется единственый файл, назначением может быть каталог:

```
FileCopy("C:\DATA.TXT", "C:\BACKUP", "Monitor");
```

При этом файл «DATA.TXT» скопируется в каталог «BACKUP» на диске «C:». Когда копирование завершится, тэг Monitor будет установлен в 1.

Если исходный файл содержит знаки шаблона, то назначением ДОЛЖЕН быть каталог (а не имя файла), иначе функция возвратит ошибку:

```
FileCopy("C:\*.TXT", "C:\BACKUP", "Monitor");
```

В этом примере копируются все файлы с расширением .TXT из корневого каталога диска C:\ в каталог C:\BACKUP. Когда это будет сделано, тэг Monitor станет равным 1.

FileDelete() системная

Удаляет ненужные или бесполезные файлы.

Синтаксис

```
FileDelete(Filename);
```

Параметр	Описание
Filename	Имя удаляемого файла.

Комментарий

Если файл успешно удален, функция возвратит 1. В противном случае функция возвратит 0.

Пример

```
Status=FileDelete("C:\DATA.TXT");
```

Status равен 1, если в корневом каталоге на диске C:\ есть файл «DATA.TXT», или 0 — если нет такого файла.

FileMove() системная

Эта функция похожа на **FileCopy()**, но в отличие от нее, она удаляет файлы в исходном расположении при переносе их из одного места в другое, а не копирует.

Синтаксис

```
FileMove(SourceFile,DestFile,DoneTag);
```

Параметр	Описание
SourceFile	Имя исходного файла (включая полный путь).
DestFile	Имя файла назначения (включая полный путь) или имя каталога (см. приведенные примеры).
DoneTag	Имя тэга, которое используется функцией **FileCopy()** для отчета о *прохождении* процедуры копирования. Этот тэг должен быть строкой, задающей имя тэга (а не сам тэг). Таким образом, если ваш тэг называется Monitor, вы должны указать «Monitor» или Monitor.Name, а не просто Monitor.

Комментарий

При использовании функции **FileMove()**, она сразу же возвращает 1, если процедура копирования успешно начала выполняться. Она возвращает 0, если в данный момент выполняется другая процедура (нельзя начать новую процедуру) или -1, если возникла ошибка. Используя возвращаемое значение, можно проследить за инициализацией функции **FileMove()**:

```
Status=FileMove("C:\DATA.TXT", "D:\DATA.TXT", "Monitor");
```

Status — это целый тэг, куда записывается 1, -1 или 0. Функция **FileMove()** выполняется в фоновом режиме, поэтому она не будет мешать выполнению InTouch. Тэг *DoneTag* позволяет приложению или пользователю отслеживать *процесс* выполнения копирования. Таким образом можно сразу привлечь внимание пользователя, если ПОСЛЕ инициализации возникли какие-либо ошибки. Это отличается от значения функции, которое возвращается в переменную *Status*, и дает результат *инициализации* функции.

Значение *DoneTag* присваивается только после успешной инициализации функции. Это значение равно 0, пока процедура находится в стадии выполнения. Оно устанавливается равным 1, если процедура выполнилась успешно, или -1, если до завершения процедуры возникла ошибка.

Если файлы *SourceFile* и *DestFile* находятся на одном и том же диске, функция может просто изменить ссылку на каталог файла (где компьютер держит имя и расположение файла на диске) без фактического переноса данных. В этом случае процесс переноса будет очень быстрым, независимо от размера файла. Если же исходный файл и файл назначения находятся на разных дисках, то время, затрачиваемое на перенос, будет зависеть от размера файла. Это потому, что данные должны быть переписаны с одного физического диска на другой.

Пример

```
FileMove ("C:\DATA.TXT","C:\BACKUP\DATA.TXT","Monitor");
```

В этом примере файл «DATA.TXT» будет перенесен из корневого каталога на диске «C» в каталог «BACKUP». Когда это завершится, тэг Monitor будет установлен в 1.

Примечание. Эту функцию можно также применять для переименования файлов, если для исходного файла и файла назначения указаны одинаковые каталоги, но разные имена.

```
FileMove ("C:\DATA.TXT","C:\DATA.BAK","Monitor");
```

Это выражение переименует файл «DATA.TXT» в «DATA.BAK» в каталоге C:\. По завершении тэг Monitor будет установлен в 1.

FileReadFields() системная

Считывает записи переменных, разделенных запятыми, (Comma Separated Values - CSV) из указанного файла.

Синтаксис

```
FileReadFields(Filename,FileOffset,StartTag,NumberOfFields);
```

Параметр	Описание
Filename	Указывает имя считываемого файла.
FileOffset	Указывает смещение в файле, с которого нужно начать чтение.
StartTag	Указывает имя тэга InTouch, куда должна быть записана первая порция данных. Имя этого тэга должно заканчиваться цифрой (напр. MyTag1). Этот тэг должен быть строкой, задающей имя тэга (а не сам тэг). Таким образом, если ваш тэг называется MyTag1, вы должны указать «MyTag1» или *MyTag1.Name*, а не просто MyTag1.
NumberOfFields	Задает число полей для считывания (число разделенных запятыми полей в каждой записи файла).

Комментарий

Если стартовый тэг - «MyTag1» и число полей равно 3, то три поля считываются из файла и записываются в MyTag1, MyTag2 и MyTag3. Эти тэги с последовательными именами должны быть предварительно созданы в системе InTouch и могут иметь различные типы (целый, текстовый и т.д.).

Примеры

Если первая строка файла C:\DATA\FILE.CSV содержит:

```
This is text, 3.1416, 5
```

Следующее выражение считает эту строку и сохранит «This is text» в MyTag1, 3.1416 в MyTag2 и 5 в MyTag3.

```
BytePosition=FileReadFields ("C:\DATA\FILE.CSV", 0, "MyTag1", 3);
```

Функция возвратит новую позицию указателя файла после чтения. Вы можете использовать это значение в качестве смещения FileOffse для следующего считывания.

```
FileReadFields ("C:\DATA\FILE.CSV",BytePosition,"MyTag1",3);
```

Примечание. Тэги текстового типа (Message) могут содержать не более 131 символа.

FileReadMessage() системная

Считывает указанное количество байт (или целую строку) из заданного файла.

Синтаксис

```
FileReadMessage(Filename,FileOffset,Message_Tag,CharsToRead);
```

Параметр	Описание
Filename	Задает файл для чтения.
FileOffset	Задает место в файле, с которого нужно начать чтение.
Message_Tag	Задает тэг для сохранения считанных данных. Сохранить можно не более 131 символа.
CharsToRead	Задает, сколько байт нужно считать из файла. При обработке текстовых файлов *CharsToRead* должен быть установлен в 0. В этом случае функция будет считывать до первого символа перевода строки (LF) в файле.

Пример

```
FileReadMessage ("C:\DATA\FILE.TXT", 0, MsgTag, 0);
```

Первая строка будет считана из файла «C:\DATA\FILE.TXT» и записана в MsgTag. Функция возвращает после считывания новую позицию указателя файла (порядковый номер байта в файле, на котором стоит указатель). При следующем чтении можно использовать возвращаемое значение для аргумента *CharsToRead*.

FileWriteFields() системная

Записывает записи переменных, разделенных запятыми, (Comma Separated Values - CSV) в указанный файл.

Синтаксис

```
FileWriteFields(Filename,FileOffset,StartTag,NumberOfFields);
```

Параметр	Описание
Filename	Задает имя файла для записи. Если файл с таким именем не существует, он будет создан.
FileOffset	Задает позицию начала записи в файле. Если FileOffset равен -1, функция будет писать в конец файла.
StartTag	Указывает имя тэга InTouch, откуда должна быть взята первая порция данных. Имя этого тэга должно заканчиваться цифрой (напр. MyTag1). Этот тэг должен быть строкой, задающей имя тэга (а не сам тэг). Таким образом, если ваш тэг называется MyTag1, вы должны указать «MyTag1» или MyTag1.Name, а не просто MyTag1.
NumberOfFields	Задает количество записываемых полей (число разделенных запятыми полей в каждой записи файла)

Комментарий

Если стартовый тэг - «MyTag1» и число полей равно 3, то три поля запишутся в файл (MyTag1, MyTag2 и MyTag3). Эти тэги с последовательными именами должны быть предварительно созданы в системе InTouch и могут иметь различные типы (целый, сообщение и т.д.).

Приведенные ниже выражения запишут строку "This is text, 3.1416, 5" в первую строку файла C:\DATA\FILE.CSV. "This is text" является текущим значением тэга MyTag1, 3.1416 - значение в MyTag2 и 5 в MyTag3:

```
FileWriteFields ("C:\DATA\FILE.CSV", 0, "MyTag1", 3);
```

Функция возвратит новую позицию указателя файла после ззаписи. Это значение можно использовать в качестве смещения *FileOffset* для следующей записи.

Следующее выражение запишет текстовую строку MyTag1 в конец файла C:\DATA\FILE.CSV:

```
FileWriteFields ("C:\DATA\FILE.CSV", -1, "MyTag1", 3);
```

FileWriteMessage() системная

Записывает указанное количество байт (или целую строку) в заданный файл.

Синтаксис

```
FileWriteMessage(Filename,FileOffset,Message_Tag,LineFeed);
```

Параметр	Описание
Filename	Задает имя файла для записи. Если файл с таким именем не существует, он будет создан.
FileOffset	Задает позицию начала чтения в файле. Если *FileOffset* равен -1, функция будет писать в конец файла.
Message_Tag	Указывает, откуда брать символы для записи в файл.
LineFeed	Указывает, добавлять или нет символы перевода строки (LF) после операции записи. При записи в текстовый файл, установите значение *LineFeed* в 1.

Комментарий

Функция возвращает после записи новую позицию указателя файла. Это значение можно использовать в качестве смещения *FileOffset* для следующей записи.

Пример

Следующее выражение запишет сообщение, содержащееся в тэге *MsgTag* в конец файла C:\DATA\FILE.TXT:

```
FileWriteMessage ("C:\DATA\FILE.TXT", -1, MsgTag, 1);
```

GetNodeName() системная

Возвращает имя узла NetDDE в строковую переменную.

Синтаксис

```
GetNodeName(Tagname,NodeNum);
```

Параметр	Описание
Tagname	Тэг текстового типа (message) будет содержать имя узла.
NodeNum	Целое число, задающее число символов для текстового тэга.

Комментарий

После выполнения этого выражения функция **GetNodeName()** считает имя локального узла и возвратит его в *Tagname*. (*NodeNum* устанавливает число символов для *Tagname*).

Примечание. Эта функция выполняется только, если запущено сетевое приложение **NetDDE** в среде Windows 95 или **Network DDE** в среде Windows NT.

Пример

```
GetNodeName("MyNodeTag",131);
If MyNodeTag == «Master» THEN
   MessageTag = «Это главная машина!»;
ENDIF;
```

GetPropertyD() GOT

Считывает дискретное значение указанного свойства в режиме исполнения.

Синтаксис

```
[ErrorNumber=]GetPropertyD("ControlName.Property",Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., *ChkBox_1* или имя объекта аларма, напр., *AlmObj_1*.
.Property	Свойство элемента управления Windows или объекта аларма. Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
Tagname	Допустимый тэг InTouch (того же типа, что и возвращаемое значение), который будет содержать значение свойства в режиме исполнения функции.

☞ Информация о кодах ошибок содержится в Приложении A.

GetPropertyI() GOT

Считывает целое значение указанного свойства в режиме исполнения.

Синтаксис

```
[ErrorNumber=]GetPropertyI("ControlName.Property",Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., ChkBox_1 или имя объекта аларма, напр., AlmObj_1
.Property	Свойство элемента управления Windows или объекта аларма. Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
Tagname	Допустимый тэг InTouch (того же типа, что и возвращаемое значение), который будет содержать значение свойства в режиме исполнения функции.

☞ Информация о кодах ошибок содержится в Приложении A.

GetPropertyM() GOT

Считывает значение типа сообщение указанного свойства в режиме исполнения.

Синтаксис

```
[ErrorNumber=]GetPropertyM("ControlName.Property",Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., ChkBox_1 или имя объекта аларма, напр., AlmObj_1
.Property	Свойство элемента управления Windows или объекта аларма. Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
Tagname	Допустимый тэг InTouch (того же типа, что и возвращаемое значение), который будет содержать значение свойства в режиме исполнения функции.

Информация о кодах ошибок содержится в Приложении А.

Hide прочие

Скрывает (делает невидимыми) различные окна из сценария. Функция **Hide** должна предшествовать имени любого окна, которое надо скрыть.

Синтаксис

```
Hide Window;
```

Параметр	Описание
Window	Название окна или текстовый тэг, содержащий это название.

Комментарий Если окно не существует во время выполнения, выражение игнорируется. Если нужно только скрыть или показать окно, то вместо этой функции лучше использовать описания анимационной связи нажатия кнопки «Показать окно» и «Скрыть окно».

Пример

```
Hide "My Popup Alarm Window";
```

HideSelf прочие

Скрывает текущее активное окно.

Синтаксис

```
HideSelf;
```

Комментарий Эту функцию можно использовать только в сценарии типа Action Script, выполняемого при «нажатии» кнопки в окне.

HTGetLastError() архивные

Определяет, произошла ли ошибка во время последнего считывания данных для указанного пера указанного архивного тренда.

Синтаксис

```
[Result=]HTGetLastError(Hist_Tag,UpdateCount,PenNum);
```

Параметр	Описание
Hist_Tag	Строка, представляющая имя архивного тренда.
UpdateCount	Целое, представляющее поле **.UpdateCount** архивного тренда.
PenNum	Целый тэг или значение, содержащий номер пера (от 1 до 8).
[Result=]	Могут быть возвращены следующие результаты:

Результат	Описание
0	Нет ошибок
1	Общая ошибка сервера
2	Старый запрос
3	Ошибка файла
4	Сервер не загружен
5	Тренд или псевдоним, переданные функции, не существуют
6	Тэг архивного тренда не существует в базе данных
7	Недопустимый номер пера, переданный функции, (не в диапазоне от 1 до 8).

Пример

Это выражение получает код ошибки последнего считывания данных для пера *Pen3* архивного тренда *Trend1* и помещает результат в целую переменную *ResultCode*.

```
[ResultCode=]HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

Следующее выражение можно использовать в анимационной связи типа «Analog Value Display» (вывод аналогового числа):

```
HTGetLastError("Trend1",Trend1.UpdateCount,3);
```

HTGetPenName() архивные

Возвращает имя тэга, используемого в текущий момент для пера с указанным номером указанного тренда.

Синтаксис

```
MessageResult=HTGetPenName(Hist_Tag,UpdateCount,PenNum);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле **.UpdateCount**.
PenNum	Целый тэг или значение, содержащий номер пера (от 1 до 8).

Возвращается текстовый тэг, содержащий имя тэга указанного пера.

Пример

Это выражение получает имя тэга для пера *Pen2* с именем тренда *Trend1* и помещает результат в текстовый тэг *TrendPen*.

```
TrendPen=HTGetPenName("Trend1",Trend1.UpdateCount,2);
```

HTGetTimeAtScooter() архивные

Возвращает время в секундах с 00:00:00 часов по Гринвичу, 1 января, 1970 года, для выборки тренда в позиции визира. Номер и позиция визира заданы аргументами *ScootNum* и *ScootLoc*.

Синтаксис

```
IntegerResult=HTGetTimeAtScooter(Hist_Tag,UpdateCount,
ScootNum,ScootLoc);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле .UpdateCount.
ScootNum	Целое число, представляющее левый или правый визир (1= левый визир, 2= правый визир).
ScootLoc	Действительное число, представляющее поле **.ScooterPosRight** или **.ScooterPosLeft**.

Комментарий Изменение любого из тэгов *UpdateCount, ScootNum* или *ScootLoc* вызывает на выполнение эту функцию. Это обеспечивает выполнение функции после новых считываний или передвижений визира.

Пример Это выражение считывает время в секундах для значения в текущей позиции левого визира при выводе тренда с именем *Trend1*:

```
HTGetTimeAtScooter("Trend1",Trend1.UpdateCount,1,
Trend1.ScooterPosLeft);
```

HTGetTimeStringAtScooter() архивные

Возвращает строку, содержащую время и дату для выборки тренда в позиции визира. Номер и позиция визира указанны аргументами *ScootNum* и *ScootLoc*.

Синтаксис

```
MessageResult=HTGetTimeStringAtScooter(Hist_Tag,
UpdateCount,ScootNum,ScootLoc,Format_Text);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле .UpdateCount
ScootNum	Целое число, представляющее левый или правый визир (1= левый визир, 2= правый визир)
ScootLoc	Действительное число, представляющее поле *.ScooterPosRight* или *.ScooterPosLeft*.
Format_Text	Строка, задающая формат вывода даты и времени. Можно задать один из следующих форматов: *"Date"*, *"Time"*, *"DateTime"*, *"DOWShort"* (день недели в сокращенной форме, напр. Wed - среда) и «DOWLong» (полное имя дня недели, напр. Wednesday - среда).

Комментарий Изменение любого из тэгов *UpdateCount*, *ScootNum* или *ScootLoc* вызывает на выполнение эту функцию. Это обеспечивает выполнение функции после новых считываний или передвижений визира. Формат строки определяет содержимое возвращаемого значения.

Пример Это выражение считывает дату и время для значения в текущей позиции правого визира при выводе тренда с именем *Trend1*. Значение сохраняется в текстовом тэге «NewRightTimeString» в формате «Time»:

```
NewRightTimeString=HTGetTimeStringAtScooter
("Trend1",Trend1.UpdateCount,2,Trend1.ScooterPosRight,"Time")
;
```

HTGetValue() архивные

Возвращает значение запрошенного типа, вычисленное для указанного пера на всем временном диапазоне тренда. После завершения считывания тренда из-за изменения поля *UpdateCount* это функция будет пересчитываться.

Синтаксис

```
RealResult=HTGetValue(Hist_Tag,UpdateCount,PenNum,ValType
_Text);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле .UpdateCount.
PenNum	Целый тэг или значение, содержащее номер пера (от 1 до 8).
ValType_Text	Строка, указывающая тип возвращаемого значения:

Тип	Описание
"PenAverageValue"	Среднее для всего тренда.
"PenMaxValue"	Максимум для всего тренда.
"PenMinValue"	Максимум для всего тренда.
"PenMaxEU"	Максимум в единицах измерения для всего тренда.
"PenMinEU"	Минимум в единицах измерения для всего тренда.
"PenStdDev"	Стандартное отклонение для всего тренда.

Примечание. При использовании параметра *ValType_Text* с функцией **HTGetValue** необходимо указывать один из перечисленных выше типов.

Комментарий Возвращаемое значение является внутренней переменной действительного типа и представляет собой вычисленное значение заданного типа.

Пример Это выражение получает стандартное отклонение для данных, считанных для пера *Pen2* тренда *Trend1*. Значение сохраняется в действительной внутренней переменной *LeftHemisphereSD*:

```
LeftHemisphereSD=HTGetValue("Trend1",Trend1.UpdateCount,
2,"PenStdDev");
```

HTGetValueAtScooter() архивные

Возвращает зничение запрошенного типа в точке, в которой указанный визир (правый или левый) пересекает граф, выводимый указанным пером указанного архивного тренда. После завершения считывания тренда из-за изменения поля *UpdateCount* это функция будет пересчитываться.

Синтаксис

```
RealResult=HTGetValueAtScooter(Hist_Tag,UpdateCount,ScootNum,
ScootLoc,PenNum,ValType_Text);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле .UpdateCount.
ScootNum	Целое число, представляющее левый или правый визир (1= левый визир, 2= правый визир)
ScootLoc	Действительное число, представляющее поле .ScooterPosRight или .ScooterPosLeft.
PenNum	Целый тэг или значение, содержащее номер пера (от 1 до 8).
ValType_Text	Строка, указывающая тип возвращаемого значения:

Тип	Описание
"PenValue"	Значение в позиции визира.
"PenValid"	0 — если значение не допустимо, или 1 — если допустимо.
	Примечание. При использовании параметра *ValType_Text* с функцией **HTGetValueAtScooter** необходимо указывать один из перечисленных выше типов.

Комментарий

Если задан тип "PenValue", функция возвращает результат во внутреннюю переменную действительного типа. Если задан тип "PenValid", функция возвращает результат во внутреннюю переменную дискретного типа.

Пример

Это выражение получает значение 1, если значение допустимо или 0, если значение недопустимо для данных, считанных для тренда *Trend1*, поля *Pen3* в текущей позиции правого визира. Значение сохраняется во внутренней переменной дискретного типа *ValidFlag*:

```
ValidFlag=HTGetValueAtScooter("Trend1",Trend1.UpdateCount,2,
Trend1.ScooterPosRight,3,"PenValid");
```

HTGetValueAtZone() архивные

Возвращает значение запрошенного типа, вычисленное для указанного пера на интервале времени между позициями левого и правого визира указанного тренда. После завершения считывания тренда из-за изменения поля *UpdateCount* эта функция будет пересчитываться.

Синтаксис

```
RealResult=HTGetValueAtZone(Hist_Tag,UpdateCount,Scoot1Loc,
Scoot2Loc,PenNum,ValType_Text);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
UpdateCount	Целое число, представляющее поле .UpdateCount.
Scoot1Loc	Действительное число, отображающее поле **.ScooterPosLeft** тэга.
Scoot2Loc	Действительное число, отображающее поле **.ScooterPosRight** тэга.
PenNum	Целый тэг или значение, содержащее номер пера (от 1 до 8).
ValType_Text	Строка, указывающая тип возвращаемого значения:

Тип	Описание
"PenAverageValue"	Среднее для зоны между правым и левым визиром.
"PenMaxValue"	Максимум для для зоны между правым и левым визиром.
"PenMinValue"	Минимум для зоны между правым и левым визиром.
"PenMaxEU"	Максимум в единицах измерения для зоны между правым и левым визиром.
"PenMinEU"	Минимум в единицах измерения для зоны между правым и левым визиром.
"PenStdDev"	Стандартное отклонение для зоны между правым и левым визиром.

Примечание. При использовании параметра *ValType_Text* с функцией **HTGetValueAtZone** необходимо указывать один из перечисленных выше типов.

Комментарий

Возвращаемое значение является действительной внутренней переменной и представляет собой вычисленное значение заданного типа. Задание значений констант *Scoot1Loc* и *Scoot2Loc* ни на что не влияет и используется только лишь для взведения признака обработки функции для целей обновления содержимого экрана. А для определения границ зоны функция пользуется полями тренда **.ScooterPosLeft** и **.ScooterPosRight** непосредственно из оперативной базы данных.

Пример

Это выражение вычисляет среднее значение для данных, считанных между правым и левым визиром для тренда «Trend1», поля *Pen1*. Значение сохраняется во внутренней переменной дискретного типа *AvgValue*:

```
AvgValue=HTGetValueAtZone("Trend1",Trend1.UpdateCount,
Trend1.ScooterPosLeft,Trend1.ScooterPosRight,1,
"PenAverageValue");
```

HTScrollLeft() архивные

Присваивает стартовому времени тренда значение, более раннее, чем текущее стартовое время, в процентах от ширины тренда. Эффектом выполнения этой функции будет прокручивание графа влево по оси времени на заданный процент.

Синтаксис

```
HTScrollLeft(Hist_Tag,Percent);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
Percent	Действительное число, содержащее процент, на который будет прокручен граф (от 0.0 до 100.0).

Пример

Это выражение прокручивает по оси времени тренд «*Trend1*» на 10%

```
HTScrollLeft("Trend1",10.0);
```

Если стартовое время тренда 12:00:00 PM и ширина тренда составляет 60 секунд, тогда стартовое время нового тренда будет 11:59:54 AM (после выполнения функции).

HTScrollRight() архивные

Присваивает стартовому времени тренда значение, более позднее чем текущее стартовое время, в процентах от ширины тренда. Эффектом выполнения этой функции будет прокручивание графа вправо по оси времени на заданный процент.

Синтаксис

```
HTScrollRight(Hist_Tag,Percent);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
Percent	Действительное число, содержащее процент, на который будет прокручен граф (от 0.0 до 100.0).

Пример

Это выражение прокручивает по оси времени тренд «*Trend1*» на 20%

```
HTScrollRight("Trend1",20.0);
```

Если стартовое время на тренде 12:00:00 PM и ширина тренда составляет 60 секунд, тогда стартовое время нового тренда будет 12:00:12 PM (после выполнения функции).

HTSelectTag() прочие

Выводит диалоговое окно **Select Tag**, в котором оператор может выбрать другой тэг для указанного пера. (В этом окне выводятся только те тэги, для которых в словаре тэгов предусмотрена архивная регистрация (включен параметр **Log Data**).

Синтаксис

```
HTSelectTag();
```

Комментарий

HTSelectTag() показывает ВСЕ тэги с включенным параметром архивной регистрации (Log Data). Однако с помощью фильтров браузера можно выводить меньший набор тэгов. Например, все имена тэгов, начинающиеся с A. Функция **HTSelectTag()** НЕ позволяет отображать всю базу данных тэгов — выводятся только архивные тэги.

Примеры

Следующее выражение может использоваться в сценарии действия по касанию кнопки. Этот сценарий откроет в WindowViewer окно браузера тэгов, в котором пользователь сможет выбрать имя тэга. Выбранный тэг будет использоваться пером 1 архивного объекта, именуемого "HistTrend".

```
HTSetPenName( "HistTrend", 1, HTSelectTag( )  );
```

См. также **HTSetPenName()**

HTSetPenName() архивные

Присваивает новое имя тэга перу тренда.

Синтаксис

```
HTSetPenName(Hist_Tag,PenNum,Tagname);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
PenNum	Целый тэг или значение, содержащее номер пера (от 1 до 8).
Tagname	Строка, задающая новое имя тэга, присваиваемое перу.

Комментарий

Эта функция предоставляет единственный способ добавить на тренд тэги от удаленных серверов архивних данных в режиме исполнения.

Примеры

В этом выражении поле *Pen3* тренда *Trend1* использует *OutletPressure* как новое имя тэга.

```
HTSetPenName("Trend1",3,"OutletPressure");
```

В этом выражении перо *TrendPen4* тренда *Trend1* использует распределенный тэг *HistPrv1.Tag1* как новое имя тэга.

```
HTSetPenName("Trend1",TrendPen4,"HistPrv1.Tag1");
```

См. также **HTSelectTag();**

HTUpdateToCurrentTime() архивные

Вызывает считывание данных и вывод их на дисплей с конечным временем, равным текущему. Стартовое время будет равно = Конечное время – Ширина графа.

Синтаксис

```
HTUpdateToCurrentTime(Hist_Tag);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.

Пример

Это выражение считывает с диска и выводит на дисплей данные для архивного тренда «*Trend1*» в текущий момент времени:

```
HTUpdateToCurrentTime("Trend1");
```

Если сейчас 3:04 и ширина дисплея составляет 60 секунд, то новое значение конечного времени будет 3:04. Новое стартовое время будет 3:03.

HTZoomIn() архивные

Вычисляет новую ширину графика и стартовое время. Если значение **.ScooterPosLeft** равно 0.0 и **.ScooterPosRight** равно 1.0, то новая ширина графика равна старой ширине, деленной на 2. Новое стартовое время вычисляется на основании величины *LockString*.

Синтаксис

```
HTZoomIn(Hist_Tag,LockString);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
LockString	Строка, задающая тип масштабирования:

Тип	Описание
"StartTime"	Оставляет стартовое время равным времени до растяжения графа.
"Center"	Оставляет центральное время равным времени до растяжения графа.
"EndTime"	Оставляет конечное время равным времени до растяжения графа.

Комментарий

Если позиции визиров не находятся на краях графика, то новая ширина графика определяется временем между значениями **.ScooterPosLeft** и **.ScooterPosRight**. В этом случае значение *LockString* не используется. Минимальная ширина графика 1 секунда. После масштабирования позиции визира будут установлены в значения **.ScooterPosLeft** = 0.0 и **.ScooterPosRight** = 1.0.

Пример

Это выражение растягивает граф (увеличивает развертку) в 2 раза и сохраняет прежнее стартовое время для тренда «Trend1». Trend1.ScooterPosRight равно 1.0, а Trend1.ScooterPosLeft равно 0.0. Если стартовое время перед растяжением было 1:25:00 и ширина графа составляла 30 секунд (перед растяжением), то стартовое время останется 1:25:00, но ширина графа станет равной 15 секундам:

```
HTZoomIn("Trend1","StartTime");
```

HTZoomOut() архивные

Вычисляет новую ширину графа и стартовое время. Новая ширина графа равна старой, умноженной на 2. Новое стартовое время вычисляется на основании величины *LockString*.

Синтаксис

```
HTZoomOut(Hist_Tag,LockString);
```

Параметр	Описание
Hist_Tag	Имя архивного тренда.
LockString	Строка, задающая тип масштабирования:

Тип	Описание
"StartTime"	Оставляет стартовое время равным времени до растяжения графа.
"Center"	Оставляет центральное время равным времени до растяжения графа.
"EndTime"	Оставляет конечное время равным времени до растяжения графа.

Комментарий

Позиции визиров никак не влияют на функцию **HTZoomOut()**. После масштабирования позиции визиров будут установлены в значения **.ScooterPosLeft** = 0.0 и **.ScooterPosRight** = 1.0.

Пример

Это выражение сжимает граф (уменьшает развертку) в 2 раза и сохраняет прежнее центральное время для тренда «*Volume*». Если стартовое время перед сжатием было 2:15:00 и ширина графа составляет 30 секунд, то стартовое время после сжатия станет 2:14:45. Ширина графа станет равной 60 секундам, а время центра графа останется 2:15:15:

```
HTZoomOut("Volume","Center");
```

InfoAppActive() системная

Проверяет, является ли приложение активным.

Синтаксис

```
DiscreteResult=InfoAppActive(AppTitle);
```

Параметр	Описание
AppTitle	Строка, содержащая название приложения.

Пример

InfoAppActive("Microsoft Excel") возвращает 1, если выполняется

InfoAppActive("Calculator") возвращает 0 {если не выполняется}

Примечание. Заголовок конкретного приложения можно определить с помощью функции **InfoAppTitle()**.

В следующем примере функция **InfoAppActive** проверяет список активных задач в системе. Если сейчас выполняется задача под названием "Notepad", будет возвращена 1. Таким образом можно избежать запуска второго экземпляра программы Notepad. Если **InfoAppActive** возвращает 0 (Notepad не выполняется), тогда программа Notepad может быть запущена.

```
IF InfoAppActive( InfoAppTitle( "Notepad" ) ) == 1 THEN
   ActivateApp  InfoAppTitle( "Notepad" );
ELSE
   StartApp  "Notepad";
ENDIF;
```

InfoAppTitle() системная

Возвращает заголовок приложения или имя в списке задач Windows для указанной программы, которая выполняется в данный момент.

Синтаксис

```
MessageResult=InfoAppTitle(«ProgramEXEName»);
```

Параметр	Описание
ProgramEXEName	Строка, содержащая название программы

Пример

Следующее выражение обработает *ProgramEXEName* со значением "calc" и возвратит "Calculator." Фактическое имя файла — calc.exe. В строке названия программы не нужно добавлять ".exe".

InfoAppTitle("calc") возвращает "Calculator"

InfoAppTitle("excel") возвращает "Microsoft Excel"

InfoDisk() системная

Возвращает информацию об указанном локальном (или сетевом) дисковом устройстве.

Синтаксис

```
IntegerResult=InfoDisk(«Drive»,InfoType,Trigger);
```

Параметр	Описание
Drive	Строка или текстовый тэг, содержащие букву диска.
InfoType	Целое, задающее тип информации:

Тип	Описание
1	Возвращает общий объем (в байтах) дискового пространства.
2	Возвращает размер свободного пространства (в байтах) на диске.
3	Возвращает общий объем (в килобайтах) дискового пространства.
4	Возвращает размер свободного пространства (в килобайтах) на диске.

Trigger	Выполняет функцию **InfoDisk()** каждый раз, когда меняется значение *Trigger*. *Trigger* может быть любым тэгом (не ограничен системными переменными). Этот параметр используется только в выражениях связей анимации, при использовании функции **InfoDisk()** в сценарии любое значение может использоваться как заглушка, поскольку данная функция не влияет на выполнение сценария.

Комментарий Информация о диске, указанном в тэге *Drive*, возвращается в тэг *IntegerResult*.

Пример Это выражение выполняется каждую минуту и возвращает текущее значение:

InfoDisk("C", 1, $Minute) возвращает 233869345 {общий объем в байтах}

InfoDisk("C", 2, $Minute) возвращает 3238935 {свободный объем в байтах}

InfoDisk("C", 3, $Minute) возвращает 228388 {общий объем в килобайтах}

InfoDisk("C", 4, $Minute) возвращает 3163 {свободный объем в килобайтах}

Комментарий

1 килобайт = 1024 байта

Примечание. Как и в других функциях, где используется единственный символ, если тэг типа сообщение, передаваемый функции InfoDisk() (буква диска), содержит более одного символа, то используется только первый символ. Поскольку эта функция использует информацию, получаемую от операционной системы, значения могут быть ошибочными в среде Windows 95 с дисками больше 2 гигабайт.

InfoFile() системная

Возвращает информацию об указанном файле или каталоге.

Синтаксис

```
IntegerResult=InfoFile(«Filename»,»InfoType»,Trigger);
```

Параметр	Описание
Filename	Строка или текстовый тэг, содержащие имя файла.
InfoType	Целое, задающее тип информации:

Тип	Описание
1	*Файл существует?* Возвращает 1, если задано имя существующего файла. Возвращает 2, если задано имя подкаталога. Возвращает 0, если функция не может найти файл.
2	размер файла (в байтах).
3	Дата и время файла (в секундах с 1 января 1970).
4	Число файлов, удовлетворяющих описанию имени файла. Значения больше 1 возвращаются, когда поиск файла осуществляется по шаблону, при котором более одного файла удовлетворяют критерию поиска.

Trigger	Выполняет функцию **InfoFile()** каждый раз, когда меняется значение *Trigger*. *Trigger* может быть любым тэгом (не ограничен системными переменными). Этот параметр используется только в выражениях связей анимации, при использовании функции **InfoFile()** в сценарии любое значение может использоваться как заглушка, поскольку данная функция не влияет на выполнение сценария.

Комментарий

Информация о файле, указанном в тэге *Filename*, возвращается в *IntegerResult*. *Filename* должен включать в себя полный путь, но может также содержать символы шаблона (*, ?).

Пример

Это выражение выполняется каждую минуту и возвращает следующие значения:

```
InfoFile("c:\IT56\view.exe", 1, $Minute) возвращает 1
{файл найден}

InfoFile("c:\InTouch\view.exe", 2, $Minute) возвращает 634960
{размер файла}

InfoFile("c:\InTouch\view.exe", 3, $Minute) возвращает
736701852
{секунд с 1-1-70}

InfoFile("c:\InTouch\*.exe", 4, $Minute) возвращает 17
{найдено 17 EXE-файлов}
```

InfoInTouchAppDir() системная

Возвращает каталог текущего приложения системы InTouch.

Синтаксис

```
MessageResult=InfoInTouchAppDir();
```

Комментарий Каталог текущего приложения будет возвращен в тэг *MessageResult*.

Пример

```
InfoInTouchAppDir() возвращает "c:\InTouch.32\demoapp1"
```

InfoResources() системная

Возвращает значения различных системных ресурсов.

Синтаксис

```
IntegerResult=InfoResources(ResourceType,Trigger);
```

Параметр	Описание
ResourceType	Целое, задающее ресурс:

Тип	Описание
1	Возвращает процент свободного пространства для ресурсов GDI.
2	Возвращает процент свободного пространства для ресурсов USER.
3	Возвращает число байт свободной оперативной памяти.
4	Возвращает число выполняемых в данный момент задач.

Trigger	Выполняет функцию **InfoResources()** каждый раз, когда меняется значение *Trigger*. *Trigger* может быть любым тэгом (не ограничен системными переменными). Этот параметр используется только в выражениях связей анимации, при использовании функции **InfoResources()** в сценарии любое значение может использоваться как заглушка, поскольку данная функция не влияет на выполнение сценария.

Комментарий Значение конкретного ресурса, указанного в тэге *ResourceType*, хранится в переменной *IntegerResult*.

Пример Это выражение выполняется каждую минуту и возвращает следующие значения:

```
InfoResources(1, $Minute) возвращает 54  {свободный %}
InfoResources(2, $Minute) возвращает 36  {свободный %}
InfoResources(3, $Minute) возвращает 11524093  {байт}
InfoResources(4, $Minute) возвращает 14  {задач}
```

Пример Значения системных ресурсов

Типы 1 и 2:
Ресурсы GDI и USER всегда возвращают '50%' под Windows NT и Windows 95.

Тип 3:
Под Windows NT и Windows 95 возвращается "количество свободных байт страничного файла".

Тип 4:
Под Windows NT и Windows 95 возвращается результат поиска всех окон верхнего уровня. Считаются только видимые окна и не имеющие владельца. Это не равно "числу выполняемых в системе задач". Ближе всего это соответствует числу задач, показанных на вкладке **Applications** в окне **Task Manager** Windows NT или в окне **Close Program**, которое открывается при нажатии CTRL+ALT+DEL в Windows 95.

Int() математическая

Возвращает ближайшее целое, меньшее или равное заданному числу.

Синтаксис

```
IntegerResult=Int(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий

При обработке отрицательного числа, функция возвратит наибольшее по модулю целое значение.

Пример

```
Int(4.7) возвращает 4
Int(-4.7) возвращает -5
```

IOSetAccessName() прочие

В среде выполнения изменяет элементы имени доступа "*application"* (приложение) и "*topic*" (тема), позволяя применять предусмотренные в InTouch технологии "горячего" резервного копирования.

Синтаксис

```
IOSetAccessName(«AccessName», «AppName», «TopicName»);
```

Параметр	Описание
AccessName	Существующее имя доступа, в котором будут меняться значения "AppName" и "Topic Name". Фактическая строка или текстовый тэг.
AppName	Новое имя приложения. Фактическая строка или текстовый тэг.
TopicName	Новое имя темы. Фактическая строка или текстовый тэг.

Комментарий

Значения "имя доступа", "имя приложения" и "имя темы" могут быть заданы в виде текстовых строк или строковых значений, полученных из других тэгов или функций InTouch. Например, имя доступа *MyAccess1* можно изменить так, чтобы оно указывало на приложение "EXCEL" и на тему «Sheet1», с помощью следующего выражения:

Пример

```
IOSetAccessName("MyAccess1", "EXCEL", "Sheet1");
```

или

```
Number = 1;
AccNameString = "MyAccess" + Text(Number, "#");
IOSetAccessName(AccNameString, "EXCEL", "Sheet1");
```

Если для темы указана пустая строка, то сохраняется текущее значение приложения, а его имя темы обновляется. Например, следующее выражение изменит в имени доступа MyAccess2 значение приложения на "excel", не меняя текущее значение темы:

```
IOSetAccessName("MyAccess2", "excel", "");
```

Таким же образом, если пустая строка указана в качестве имени приложения, в этом случае текущее значение темы сохраняется, а обновляется имя приложения. Например, следующее выражение изменит в тэге MyAccess3 значение темы на "Sheet2", не меняя текущее имя приложения:

```
IOSetAccessName("MyAccess3", "", "Sheet2");
```

Примечание. При обработке функции **IOSetAccessName()** возникает некоторая задержка, поскольку текущий диалог прерывается, и начинается новый. В этот момент любые операции POKE или попытки записи новой темы будут потеряны.

IOSetItem() прочие

Изменяет имя доступа и(или) элемента в поле **.Reference** внешнего тэга.

Синтаксис

```
IOSetItem(«Tagname»,»AccessName»,»Item»);
```

Параметр	Описание
Tagname	Любой внешний тэг в кавычках.
AccessName	Имя доступа, которое нужно установить в тэге.
Item	Имя элемента, которое нужно установить в тэге.

Примеры

```
IOSetItem(TagName, AccessName, Item)
```

Значения Tagname, AccessName и Item могут быть заданы в виде текстовых строк или строковых значений, полученных из других тэгов или функций InTouch. Например, поле **.Reference** тэга "MyTag1" можно изменить так, чтобы оно указывало на имя доступа "excel" и элемент "R1C1", с помощью следующего выражения:

IOSetItem("MyTag1", "excel", "R1C1");

или

Number = 1;
TagNameString = "MyTag" + Text(Number, "#");
IOSetItem(TagNameString, "excel", "R1C1");

Если пустая строка ("") задана для обоих значений, то этот тэг будет деактивирован. Например, следующее выражение деактивирует тэг MyTag2:

IOSetItem("MyTag2", "", "");

Если пустая строка задана только для элемента, то его значение сохраняется, а значение имя доступа обновляется. Например, следующее выражение изменит имя доступа тэга MyTag3 на "excel2", не меняя существующее значение элемента:

IOSetItem("MyTag3", "excel2", "");

Таким же образом, если пустая строка задана только для имени доступа, то текущее значение элемента сохраняется, а имя доступа обновляется. Например, следующее выражение изменит значение элемента тэга MyTag3 на "R1C2", не меняя текущее значение имени доступа:

IOSetItem("MyTag4", "", "R1C2");

IsAnyAsynchFunctionBusy() системная

Производит проверку на наличие выполняемых асинхронных Quick-функций. Эта функция позволяет создать Quick-сценарий, который заставляет одну асинхронную Quick-функцию ждать завершения всех других выполняемых Quick-функций. Таким образом происходит ресинхронизация Quick-сценария.

Синтаксис

```
DiscreteTag=IsAnyAsynchFunctionBusy(timeout);
```

Параметр	Описание
DiscreteTag	Дисктерный тэг, в который возвращаются следующие значения: Если асинхронные функции выполняются, и период ожидания их завершения истекает, то в тэг возвращается значение 1 (истинно). Если никакие асинхронные Quick-функции не выполняются, немедленно возвращается 0 (ложно), или Quick-функция ждет истечения тайм-аута. Значение 0 будет также возвращено, если никакие другие асинхронные функции не будут выполняться по истечении периода ожидания.
timeout	Целое значение, содержащее количество секунд для ожидания результатов проверки на наличие выполняемых асинхронных функций.

Пример

Пердположим, что треубется установить соединение с разными базами SQL через асинхронные Quick-функции и при этом известно, что установка этих соединений будет длиться 2 минуты. Прежде всего, мы выполняем асинхронную Quick-функцию для соединения с базами SQL. Затем выполняем функцию **IsAnyAsynchFunctionBusy(120)** для выделения периода ожидания, достаточного для того, чтобы соединения были установлены до завершения Quick-функции.

Однако, если через 2 минуты соединения не будут установлены, и асинхронные Quick-функции будут все еще пытаться их установить, в функцию **IsAnyAsynchFunctionBusy()** возвращается значение 1 (истинно). После этого можно выводить сообщение об ошибке, информирующее оператора о неудачной попытке соединения SQL.

Можно использовать следующий Quick-сценарий окна **On Show**:

```
IF IsAnyAsynchFunctionBusy(120) == 1 THEN
   SHOW "SQL Connection Error Dialog";
ENDIF;
```

Log() математическая

Возвращает натуральный логарифм числа.

Синтаксис

```
RealResult=Log(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий

Вычисляет натуральный логарифм числа *Number* и результат возвращает в переменную *Result*.

Пример

```
Log(100) возвращает 4.605...
Log(1) возвращает 0
```

LogMessage прочие

Записывает определенное пользователем сообщение в журнал Wonderware Logger.

Синтаксис

```
LogMessage(«Message_Tag»);
```

Параметр	Описание
Message_Tag	Строка или текстовый тэг для записи в журнал.

Комментарий

Эта функция очень полезна для отладки сценариев InTouch. Путем правильной расстановки функций **LogMessage()** в сценарии можно определить порядок выполнения сценария, производительность сценария и идентифицировать значения тэгов до и после их обработки сценарием. Каждое сообщение заносится в журнал Wonderware Logger с точными метками времени и даты.

Пример

```
LogMessage("Report Script is Running");
```

Вышеуказанное выражение занесет в журнал Wonderware Logger следующую запись:
94/01/14 15:21:14 WWSCRIPT Message:Report Script is Running.

```
LogMessage(«The Value of MyTag is « + Text(MyTag, «#»));
MyTag+MyTag + 10;
LogMessage(«The Value of MyTag is « + Text(MyTag, «#»));
```

LogN() математическая

Возвращает значение логарифма от *x* по основанию *n*.

Синтаксис

```
Result=LogN(Number,Base);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.
Base	Целое число, устанавливающее *основание* логарифма.

Комментарий

При основании 1 результат не определен.

Пример

```
LogN(8, 3) возвращает 1.89279...
LogN(NumberTag, BaseTag) возвращает 0.564...если NumberTag
содержит 3, а BaseTag содержит 7.
```

Pi() математическая

Возвращает значение *P* (Пи).

Синтаксис

```
RealResult=Pi();
```

Пример

```
Pi() возвращает 3.1415926...
```

PlaySound() прочие

Выдает звуковой сигнал, заданный в файле .wav или в разделе **[Sounds]** файла WIN.INI через звуковое устройство Windows (если установлено).

Синтаксис

```
PlaySound(«SoundName»,Flags);
```

Параметр	Описание
SoundName	Строка или текстовый тэг, содержащие имя звукового файла.
Flags	*Флаги* могут принимать одно из следующих значений:

Тип	Описание
0	Выдает звук синхронно (по умолчанию).
1	Выдает звук асинхронно.
2	Не использует звук по умолчанию. Звуковой файл имеет расширение .WAV. Функция принимает также на вход строку из раздела [Sounds] в файле WIN.INI.
3	НЕ ИСПОЛЬЗУЕТСЯ!
4	Ссылка на файл в памяти. Этот флаг не надо использовать в сценариях InTouch.
5-7	НЕ ИСПОЛЬЗУЕТСЯ!
8	Повторяет звук до следующего вызова функции **PlaySound()**.

Пример

```
PlaySound ("c:\horns.wav",1);
```

Примечание. Звук должен находиться в имеющейся физической памяти, и должна быть техническая возможность проигрывать его на установленном звуковом устройстве. Каталоги для поиска звуковых файлов просматриваются в таком порядке: текущий каталог; каталог Windows; системный каталог windows\system; каталоги, перечисленные в PATH. Если не удается найти указанный .wav файл, проигрывается файл, указанный в пункте "Стандартный звук" (**Настройка**, **Панель управления**, вкладка **Звук**). Если нельзя найти и этот файл, не выдается никакого звука.

PrintHT() архивные

Создает кнопку, чтобы использовать ее для печати графа архивного тренда, связанного с указанным тэгом типа Архивного тренда. Использовать эту функцию следует тогда, когда архивный тренд видим на экране.

Синтаксис

```
PrintHT("Trend_Tag");
```

Параметр	Описание
Trend_Tag	Строка или текстовый тэг, содержащий имя архивного тренда.

Комментарий Отсутствует

Пример

```
PrintHT("HistTrend1");
```

PrintWindow() прочие

Печатает указанное окно.

Синтаксис

```
PrintWindow("Window",Left,Top,Width,Height,Options);
```

Параметр	Описание
Window	Имя окна, которое надо напечатать — строка или текстовый тэг.
Left	Число с плавающей точкой, задаваемое в дюймах для установки левого поля (отступа).
Top	Число с плавающей точкой, задаваемое в дюймах для установки верхнего поля (отступа).
Width	Число с плавающей точкой, задаваемое в дюймах для установки ширины печатной копии. Оно может быть установлено равным заданной ширине или равно 0, тогда по умолчанию принимается наибольший коэффициент соотношения.
Height	Число с плавающей точкой, задаваемое в дюймах для установки высоты печатной копии. Оно может быть установлено равным заданной высоте или равно 0, тогда по умолчанию принимается наибольший коэффициент соотношения.
Options	Дискретное значение, 0 или 1, используется только, если *Width* и *Height* равны 0. Если *Options*=1, окно печатается с наибольшим коэффициентом соотношения, кратным размеру окна. Если *Options*=1, окно распечатывается с наибольшим коэффициентом соотношения, чтобы уместилось на бумагу.

Примечание. Если окно содержит растровое изображение, установите *Options* в 1, чтобы не растягивать растры.

Комментарий

Используя эту функцию, можно поставить в очередь много отчетов. (Рекомендуется использовать функцию **PrintWindow()** вместо функции **PrintHT()**. Пользователю рекомендуется инициировать печать из диалога архивного тренда, если нужно напечатать целый экран, а не только граф.)

Во время выполнения этой функции программа View загружает окно во «внеэкранную» область памяти. Затем View выжидает 10 секунд для обновления всех переменных DDE (динамический обмен данными). Потом окно посылается на принтер. Количество времени выжидания View можно регулировать добавлением следующей строки в файл INTOUCH.INI:

```
PrintWindowWait=10000
```

10000 представляет собой число миллисекунд на ожидание.

Шрифты печатаются как шрифты, объекты отображаются в растры и печатаются как растровые изображения. Окна с белым фоном, содержащие только шрифты, печатаются очень быстро. Окна с цветным фоном, содержащие много объектов, займут для печати больше времени.

Примечание. Чтобы обеспечить правильную печать текста в окне, рекомендуется во всех окнах для текстовых полей использовать шрифты «True Type».

При печати кнопок текст на кнопке может оказаться «обрезанным», потому что для печати текста на кнопке используется шрифт «System», который не является масштабируемым шрифтом «True Type». А шрифт «System» при печати выглядит не совсем так, как на экране. В этом случае попробуйте расширить кнопку.

Если принтер используется для печати алармов, то для функции **PrintWindow()** понадобится другой принтер.

Примеры В следующем сценарии, выполняемом по условию Истина (On True Condition Script), печатаются три страницы отчета каждый день в 8:30 утра:

Условия:
```
$Hour == 8 AND $Minute == 30
```

Сценарий:
```
PrintWindow("Сводка по 1-й смене",1,1,0,0,0);
PrintWindow("Сводка по 2-й смене",1,1,0,0,0);
PrintWindow("Сводка по 3-й смене",1,1,0,0,0);
```

Функция **PrintWindow()** возвращает 1, если имя окна существует и может быть поставлено в очередь на печать. В противном случае, она возвращает 0. Таким образом можно отследить статус функции.

```
Status=PrintWindow("Сводка по смене",1,1,0,0,0);
```

Status — это дискретный тэг, в который записывается 1 или 0.

RecipeDelete() рецепты

Удаляет определенные на данный момент имена рецептов из указанного файла рецептов.

Синтаксис
```
RecipeDelete("Filename","RecipeName");
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле рецептов, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта *RecipeName*. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой или текстовым тэгом.

Пример Это выражение удаляет рецепт «Recipe1» из файла recfile.csv:

```
RecipeDelete("c:\recipe\recfile.csv", "Recipe1");
```

RecipeGetMessage() рецепты

Записывает код ошибки выполнения функции в аналоговый тэг и соответствующее сообщение об ошибке в текстовый тэг.

Синтаксис

```
RecipeGetMessage(Analog_Tag,Message_Tag,Number);
```

Параметр	Описание
Analog_Tag	Аналоговый целый или действительный тэг для кода ошибки.
Message_Tag	Текстовый тэг, содержащий сообщение об ошибке.
Number	Если функция должна заполнить параметр символами, это поле устанавливает максимальную длину строки, возвращаемой тэгу. В системе InTouch текстовые тэги имеют максимальную длину 131 символ. Используйте 131 символ для этого тэга, если вы не сократили максимальную длину строки тэга InTouch. Этот параметр может быть константой или аналоговым тэгом.

Пример

Используя функцию **RecipeGetMessage()** в сценарии изменения данных (Data Change Script) можно записать соответствующий код ошибки в аналоговый тэг, а соотвествующее сообщение об ошибке в текстовый тэг:

```
Data Change Script Tagname[.field]:ErrorCode

Script body:RecipeGetMessage(ErrorCode, ErrorMessage,131);
```

Это выражение будет автоматически выполняться всегда, когда будет меняться значение аналогового тэга ErrorCode. Когда оно изменится, функция **RecipeGetMessage()** считает текущее числовое значение тэга ErrorCode и возвратит сообщение, связанное с этим значением в тэг ErrorMessage.

```
ErrorCode = RecipeLoad
("c:\App\recipe.csv","Unit1","cookies");

RecipeGetMessage(ErrorCode, ErrorMessageTag, 131);
```

RecipeLoad() рецепты

Загружает указанный рецепт в указанный блок тэгов.

Синтаксис

```
RecipeLoad("Filename","UnitName","RecipeName");
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
UnitName	Имя указанного блока в заданном файле *шаблона рецептов*, который будет использоваться функцией. Функция **RecipeLoad()** требует указания *UnitName*. Функция **RecipeSelectUnit()** возвращает значение в этот тэг. *UnitName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле *шаблона рецептов*, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта RecipeName. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой без кавычек или текстовым тэгом.

Пример

Это выражение загружает значения, определенные для рецепта по имени Recipe1 (в файле recfile.csv), в тэги, определенные для указанного блока:

```
RecipeLoad("c:\recipe\recfile.csv", "Unit1", "Recipe1");
```

RecipeSave() рецепты

Сохраняет вновь созданный рецепт или сохраняет сделанные изменения в существующем рецепте в указанный файл рецептов.

Синтаксис

```
RecipeSave("Filename","UnitName","RecipeName");
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
UnitName	Имя указанного блока в заданном файле рецептов, который будет использоваться функцией. Функция **RecipeLoad()** требует указания *UnitName*. Функция **RecipeSelectUnit()** возвращает значение в этот тэг. *UnitName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле рецептов, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта RecipeName. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой без кавычек или текстовым тэгом.

Пример

При выполнении этого выражения все изменения, сделанные в рецепте по имени Recipe3, сохраняются в файле recfile.csv. Если Recipe3 еще не существует в файле recfile.csv, он будет создан и его значения будут присвоены тэгам, определенным в блоке Unit2:

```
RecipeSave("c:\recipe\recfile.csv", "Unit2", "Recipe3");
```

RecipeSelectNextRecipe() рецепты

Выбирает имя следующего рецепта, определенного в текущий момент в файле рецептов.

Синтаксис

```
RecipeSelectNextRecipe("Filename", RecipeName, Number);
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле рецептов, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта *RecipeName*. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой без кавычек или текстовым тэгом.
Number	Если функция должна заполнить параметр символами, это поле устанавливает максимальную длину строки, возвращаемую тэгу. В системе InTouch строковые тэги (сообщения) имеют максимальную длину 131 символ. Используйте 131 символ для этого тэга, если вы не сократили максимальную длину строки тэга InTouch. Этот параметр может быть константой или целым тэгом.

Пример

Это выражение заставляет систему считать текущее значение тэга *RecipeName* и возвратить следующий рецепт в файле. Если значение *RecipeName* пусто или не может быть найдено, возвращается первый рецепт в файле. Если *RecipeName* в данный момент содержит имя последнего рецепта в файле, оно возвращается неизменным. (Рецепты хранятся в том порядке, как они создавались.)

```
RecipeSelectNextRecipe("c:\recipe\recfile.csv", RecipeName,
131);
```

RecipeSelectPreviousRecipe() рецепты

Выбирает имя предыдущего рецепта, определенного в текущий момент в файле рецептов.

Синтаксис

```
RecipeSelectPreviousRecipe("Filename",RecipeName,Number);
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле *шаблона рецептов*, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта RecipeName. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой без кавычек или текстовым тэгом.
Number	Если функция должна заполнить параметр символами, это поле устанавливает максимальную длину строки, возвращаемую тэгу. В системе InTouch строковые тэги (сообщения) имеют максимальную длину 131 символ. Используйте 131 символ для этого тэга, если вы не сократили максимальную длину строки тэга InTouch. Этот параметр может быть константой или целым тэгом.

Пример

Это выражение заставляет систему считать текущее значение тэга *RecipeName* и возвратить предыдущий рецепт в файле. Если значение *RecipeName* пусто или не может быть найдено, возвращается первый рецепт в файле. Если *RecipeName* в данный момент содержит имя первого рецепта в файле, оно возвращается неизменным. (Рецепты хранятся в том порядке, как они создавались.)

```
RecipeSelectPreviousRecipe("c:\recipe\recfile.csv",
RecipeName,
131);
```

RecipeSelectRecipe() рецепты

Выбирает указанное имя рецепта, определенного в текущий момент в файле рецептов.

Синтаксис

```
RecipeSelectRecipe("Filename",RecipeName,Number);
```

Параметр	Описание
FileName	Имя файла *шаблона рецептов*, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
RecipeName	Указанный рецепт в назначенном файле рецептов, который будет использоваться в функции. Функции **RecipeLoad()**, **RecipeSave()** и **RecipeDelete()** требуют задать им имя рецепта *RecipeName*. Функция **RecipeSelectRecipe()** возвращает значение для этого тэга. *RecipeName* может быть строковой константой без кавычек или текстовым тэгом.
Number	Если функция должна заполнить параметр символами, это поле устанавливает максимальную длину строки, возвращаемую тэгу. В системе InTouch строковые тэги (сообщения) имеют максимальную длину 131 символ. Используйте 131 символ для этого тэга, если вы не сократили максимальную длину строки тэга InTouch. Этот параметр может быть константой или целым тэгом.

Пример

Это выражение открывает на экране окно диалога выбора рецепта (**Select a Recipe**):

```
RecipeSelectRecipe("c:\recipe\recfile.csv", RecipeName, 131);
```

Когда рецепт выбран из диалогового окна, его имя передается в тэг *RecipeName*.

RecipeSelectUnit() рецепты

Выбирает блок тэгов, в которые будут загружены значения текущего рецепта.

Синтаксис

```
RecipeSelectUnit("Filename", UnitName,Number);
```

Параметр	Описание
FileName	Имя файла шаблона рецептов, с которым будет работать функция. *FileName* может быть строковой константой или текстовым тэгом.
UnitName	Имя указанного блока в заданном файле рецептов, который будет использоваться функцией. Функция **RecipeLoad()** требует задания *UnitName*. Функция **RecipeSelectUnit()** возвращает значение в этот тэг. UnitName может быть строковой константой без кавычек или текстовым тэгом.
Number	Если функция должна заполнить параметр символами, это поле устанавливает максимальную длину строки, возвращаемую тэгу. В системе InTouch строковые тэги (сообщения) имеют максимальную длину 131 символ. Используйте 131 символ для этого тэга, если вы не сократили максимальную длину строки тэга InTouch. Этот параметр может быть константой или целым тэгом.

Пример

Это выражение открывает на экране окно диалога выбора блока (**Select a Unit**):

```
RecipeSelectUnit("c:\recipe\recfile.csv", UnitName, 131);
```

Когда блок выбран из диалогового окна, его имя передается в тэг *UnitName*.

Примечание. Обе функции **RecipeSelectRecipe()** и **RecipeSelectUnit()** используются в сочетании с функцией **RecipeLoad()**.

📖 Подробная информация о комбинировании функций рецептов содержится в разделе "Комбинирование функций рецептов" в "*Руководстве пользователя Recipe Manager*".

RestartWindowViewer системная

Позволяет пользователю управлять закрытием и перезапуском WindowViewer.

Синтаксис

```
RestartWindowViewer;
```

Комментарий

Эта функция закроет, а затем автоматически откроет Window Viewer. Она используется для обновления приложения тогда, когда не используются функции автоматического обновления сетевых приложений (NAD - Network Application Development). Эту функцию можно использовать вместе с тэгом **$ApplicationChanged** для того, чтобы определить, было ли обновление сетевого приложения, и затем осуществить пользовательское закрытие Window Viewer.

См. также **$ApplicationChanged**

Round() математическая

Округляет указанное действительное число с указанной точностью.

Синтаксис

```
RealResult=Round(Number,Precision);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.
Precision	Устанавливает точность, с которой будет округлено число.

Комментарий

Параметр *Precision* устанвливает точность, с которой будет округлено число *Number*.

Примеры

```
Round(4.3, 1) возвращает 4
Round(4.3, .01) возвращает 4.30
Round(4.5, 1) возвращает 5
Round(-4.5, 1) возвращает -5
Round(106, 5) возвращает 105
Round(43.7, .5) возвращает 43.5
```

См. также **Trunc()**

SendKeys прочие

Посылает коды клавиш другому приложению. Для другого приложения будет казаться, что клавиши были нажаты с клавиатуры. Эту возможность можно применять для ввода данных в приложение или выдачи команд приложению. В операторе SendKeys могут использоваться большинство клавиш клавиатуры. Каждая клавиша представлена одним или более символом, например A для буквы A или {ENTER} для клавиши Enter.

Синтаксис

SendKeys *KeySequence*;

Параметр	Описание
KeySequence	Любая последовательность клавиш в текстовом тэге.

Комментарий

Для того, чтобы указать более одной клавиши, перечислите подряд коды для каждого символа. Например, чтобы указать знак доллара ($), а за ним (b), введите $b. Ниже перечислены коды клавиш, однозначно соответствующие клавишам клавиатуры, которые можно посылать другим приложениям:

Клавиша	Код	Клавиша	Код
BACKSPACE	{BACKSPACE} или {BS}	HOME	{HOME}
BREAK	{BREAK}	INSERT	{INSERT}
CAPSLOCK	{CAPSLOCK}	LEFT	{LEFT}
DELETE	{DELETE} или {DEL}	NUMLOCK	{NUMLOCK}
DOWN	{DOWN}	PAGE DOWN	{PGDN}
END	{END}	PAGE UP	{PGUP}
ENTER	{ENTER} или ~ (tilde)	PRTSC	{PRTSC}
ESCAPE	{ESCAPE} или {ESC}	RIGHT	{RIGHT}
F1	{F1}*	TAB	{TAB}
		UP	{UP}

* Все функциональные клавиши вводятся аналогично.

Для специальных символов (SHIFT, CTRL и ALT) существуют собственные коды:

Клавиша	Код
SHIFT	+ (плюс)
CTRL	^ (каретка)
ALT	% (процент)

Примеры

Если нужно использовать вместе две или более клавиши, требуется вторая пара скобок. Следующее выражение используется для удержания клавиши CTRL во время нажатия клавиши ALT, а затем p:

```
SendKeys "^(%(p))";
```

Командам может предшествовать команда **ActiveApp** для того, чтобы направить нажатие клавиш нужному приложению.

Следующее выражение передает фокус приложению Excel и посылает комбинацию клавиш CTRL+P (которая может запустить на выполнение ранее определенный макрос, связанный с комбинацией клавиш CTRL+P):

```
ActivateApp "Microsoft Excel";
SendKeys "^(p)";
```

А следующее выражение вызовет на экран WindowViewer диалоговое окно входа пользователя:

```
SendKeys "%(SYL)";
```

Для кнопки вызова справки можно использовать такой сценарий действия:

```
SendKeys "{F1}";
```

SetDDEAppTopic() прочие

Эта функция заменена на **IOSetAccessName**, начиная с InTouch версии 7.0.

См. описание функции **IOSetAccessName** в этом руководстве.

SetDDEItem() прочие

Эта функция заменена на **IOSetItem**, начиная с InTouch версии 7.0.

См. описание функции **IOSetItem** в этом руководстве.

SetPropertyD() GOT

Задает дискретное значение свойства, которое должно быть записано в режиме исполнения.

Синтаксис

```
[ErrorNumber=]SetPropertyD("ControlName.Property", DiscreteTag);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., ChkBox_1 или имя объекта аларма, напр., AlmObj_1
.Property	Свойство элемента управления Windows или объекта аларма. ☞ Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
DiscreteTag	Дискретный тэг, содержащий значение, которое надо записать в режиме исполнения функции. Типичные значения: 0= Элемент управления отключен 1= Элемент управления включен

☞ Информация о кодах ошибок содержится в Приложении A.

SetPropertyI() GOT

Задает целое значение свойства, которое должно быть записано в режиме исполнения.

Синтаксис

```
[ErrorNumber=]SetPropertyI("ControlName.Property", Integer);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., ChkBox_1 или имя объекта аларма, напр., AlmObj_1
.Property	Свойство элемента управления Windows или объекта аларма. ☞ Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
Integer	Константа или целый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

SetPropertyM() GOT

Задает значение свойства текстового (message) тэга, которое должно быть записано в режиме исполнения.

Синтаксис

```
[ErrorNumber=]SetPropertyM("ControlName.Property",
"MessageTag");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, напр., ChkBox_1 или имя объекта аларма, напр., AlmObj_1
.Property	Свойство элемента управления Windows или объекта аларма. ☞ Более подробная информация об этих свойствах содержится в главе 2, *"Поля тэгов"*.
MessageTag	Тэг текстового типа, содержащий значение, которое нужно записать в режиме исполнения функции.

☞ Информация о кодах ошибок содержится в Приложении A.

Sgn() математическая

Определяет знак заданного числа (положительный, ноль или отрицательный).

Синтаксис

```
IntegerResult=Sgn(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Если входное число положительное, результат будет 1. Отрицательные числа дадут результат -1, а 0 возвратит 0.

Примеры

Sgn(425) возвратит 1

Sgn(0) возвратит 0

Sgn(-37.3) возвратит -1

Show прочие

Выводит на дисплей указанное окно (заголовок окна должен быть в кавычках).

Синтаксис

```
Show "Window";
```

Параметр	Описание
Window	Имя выводимого окна — строка или текстовый тэг.

Комментарий

Window должно соответствовать имени существующего окна или создаваемого окна. Если окно не существует в режиме исполнения, Window Viewer проигнорирует эту функцию. Если название окна поменялось, его необходимо изменить и в файле сценария.

Пример

```
Show "Alarm Summary Window";
```

Примечание. Если сценарий только скрывает или показывает окно, рекомендуется использовать анимационные связи **Show Window** или **Hide Window**. В этом случае при изменении имени окна система InTouch автоматически сделает изменение.

ShowAt() прочие

Устанавливает вертикальные и горизонтальные координаты окна в пикселах при выводе на экран.

Синтаксис

```
ShowAt("Window",Horiz,Vert);
```

Параметр	Описание
Window	Имя окна — строка или текстовый тэг.
Horiz	Горизонтальная координата в пикселах — константа или целый тэг.
Vert	Вертикальная координата в пикселах — константа или целый тэг.

Комментарий

Когда окно откроется, оно будет отцентрировано по горизонтальной и вертикальной координате. Окно не будет отцентрировано, если один из его краев будет уходить за экран. Window Viewer сам подберет положение для этого края окна.

Примеры

В этом примере горизонтальная пиксельная координата равна 100, а вертикальная пиксельная координата равна 200:

```
ShowAt("Window Name",100,200);
```

Могут быть также созданы графические объекты аналогового ввода и связаны анимационной связью с внутренними тэгами, например, *TagHoriz* и *TagVert* для динамического изменения местоположения окна в режиме исполнения. В этом случае нужно ввести следующее выражение:

```
ShowAt("Boiler Room 7 Details",TagHoriz,TagVert);
```

В следующем примере определяются пиксельные координаты объекта. Внутренние тэги **$ObjHor** и **$ObjVer** могут быть присвоены связям типа «Analog Output» (аналоговые выходы), которые дают значения координат выбранного в данный момент объекта. Можно заставить окно появиться отцентрированным по верху объекта или кнопки, используя тэги **$ObjHor** и **$ObjVer** в сценарии, связанном с этим объектом или кнопкой.

```
ShowAt("Window Name",$ObjHor,$ObjVer);
```

См. также **$ObjHor, $ObjVer, ShowTopLeftAt()**

ShowHome прочие

Выводит на дисплей «домашнее» окно (окна). «Домашними» являются окна, которые были сконфигурированы в экране свойств **WindowViewer Properties - Home Windows**.)

Синтаксис

```
ShowHome;
```

ShowTopLeftAt() прочие

Устанавливает вертикальные и горизонтальные координаты (в пикселах) верхнего левого угла окна при его выводе на экран.

Синтаксис

```
ShowTopLeftAt("Window",Horiz,Vert);
```

Параметр	Описание
Window	Имя окна. Фактическая строка или текстовый тэг.
Horiz	Горизонтальная координата в пикселах.
Vert	Вертикальная координата в пикселах.

Комментарий Когда окно будет открываться, его левый верхний угол будет размещен по указанным горизонтальной и вертикальной координатам. (Пиксельные координаты самого левого верхнего угла экрана равны 0,0). Эта функция работает так же, как функция **ShowAt()**, за исключением того, что она управляет расположением только левого верхнего угла окна.

См. также **ShowAT()**

Sin() математическая

Возвращает *синус* угла, заданного в градусах.

Синтаксис

```
Result=Sin(AngleNumber);
```

Параметр	Описание
AngleNumber	Значение угла в градусах. Любое число или тэг действительного или целого типа.

Комментарий Вычисляется значение синуса числа *Number*, и результат возвращается в переменную *Result*.

Пример

```
Sin(90) возвращает 1
Sin(0) возвращает 0
wave = 100 * sin (6 * $second);
```

См. также **Cos(), Tan()**

SPCConnect() SPC

Эта функция взаимодействует с наборами автоматического сбора данных. Прежде чем набор автоматического сбора данных начнет собирать данные, необходимо выполнить эту функцию, чтобы указать SPC, какой пользователь работает на узле.

Синтаксис

```
SPCConnect(«User»,»Password»);
```

Параметр	Описание
User	Имя пользователя базы данных. Фактическая строка или текстовый тэг.
Password	Пароль пользователя. Фактическая строка или текстовый тэг.

Комментарий Эта функция подключает пользователя к базе данных и запускает автосбор всех наборов данных с использованием идентификатора пользователя. Если для базы данных не требуется пароль, можно использовать следующий сценарий.

Пример

```
SPCConnect(«User1», «»);
```

SPCDisconnect() SPC

Эта функция отсоединяет Агента от базы данных SPC Pro. При ее выполнении останавливается автосбор всех наборов данных, связанных с Агентом.

Синтаксис

```
SPCDisconnect();
```

Комментарий При выполнении этой функции Агент отсоединяется от базы данных, и останавливается автосбор всех наборов данных, связанных с Агентом.

Пример

```
SPCDisconnect();
```

SPCDisplayData() SPC

Эта функция позволяет удобно прокручивать граф до любой даты или времени. Можно использовать тэг для отслеживания статуса поиска данных в SPC. Статус будет равным **0,** если в SPC найдены данные, или равным **1**, если данные за указанный период не найдены.

Синтаксис

```
[Status=]SPCDisplayData("Dataset", "DateString",
"TimeString", RangeInHours );
```

Параметр	Описание
Dataset	Фактическое имя набора данных — строка или текстовый тэг.
DateString	Дата в формате мм/дд/гг — строка или текстовый тэг.
TimeString	Время в формате чч:мм:сс — строка или текстовый тэг.
RangeInHours	Часы отображаемых данных — константа или целый тэг.

Пример

```
StatusTag = SPCDisplayData("Dataset", "DateString",
"TimeString" , RangeInHours);
```

SPCLocateScooter() SPC

Эта функция используется для удобства прокрутки визира к требуемому номеру выборки. Тэг визира, определенный в наборе данных, принимает значение выборки X-Bar. Установка номера выборки (SampleNumber) в значение **0** скрывает/отключает визир.

Синтаксис

```
SPCLocateScooter( "Dataset", SampleNumber );
```

Параметр	Описание
Dataset	Фактическое имя набора данных — строка или текстовый тэг.
SampleNumber	Номер выборки — константа или целый тэг.

SPCMoveScooter() SPC

Эта функция используется для удобства прокрутки визира к требуемому номеру выборки. Тэг визира, определенный в наборе данных, принимает значение выборки X-Bar.

Синтаксис

```
SPCMoveScooter( "Dataset", IncrementValue );
```

Параметр	Описание
Dataset	Фактическое имя набора данных — строка или текстовый тэг.
IncrementValue	Любое число приращения. Положительное прокручивает вперед, отрицательное — назад. Константа или целый тэг.

SPCSaveSample() SPC

Сохраняет значение выборки, введенной вручную. Эта функция используется совместно с функцией **SPCSetMeasurement()**.

Синтаксис

```
SPCSaveSample("Dataset");
```

Параметр	Описание
Dataset	Имя набора данных — строка или текстовый тэг.

Комментарий В результате выполнения этой функции данные будут введены в указанный набор данных. Функция будет использовать текущие значения переменных ручного ввода **MI_Mx** (x=номер измерения от 1 до 25) для измерений в выборке. Значения **MI_Mx** устанавливаются либо через DDE, либо с помощью функции **SPCSetMeasurement()**.

См. также **SPCSetMeasurement()**

SPCSelectDataset() SPC

Предоставляет пользователю возможность выбрать непосредственный набор данных.

Синтаксис

```
DatasetName=SPCSelectDataset()
```

Комментарий

Эта функция открывает на экране диалоговое окно **Select a Dataset**.

Когда имя набора данных выбрано, функция записывает его в тэг *DatasetName*. Этот выбор можно также использовать для изменения тэга *DatasetName* для косвенного набора данных (Indirect Dataset).

SPCSelectProduct() SPC

Предоставляет пользователю возможность выбрать продукт в заданном наборе данных.

Синтаксис

```
ProductName=SPCSelectProduct(Dataset);
```

Комментарий

Эта функция открывает на экране диалоговое окно **Select a Product**.

Когда имя продукта выбрано, функция записывает его в тэг *ProductName*. Этот имя можно теперь использовать для изменения накопленного продукта в наборе данных.

SPCSetControlLimits() SPC

Предоставляет удобный способ для ручного или автоматического ввода значений пределов управления для контрольных графов (Control Chart).

Синтаксис

```
SPCSetControlLimits("Dataset",XUCL,XLCL);
```

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
XUCL	Представляет значение UCL графа (верхний предел управления). Константа или действительный тэг.
XLCL	Представляет значение LCL графа (нижний предел управления). Константа или действительный тэг.

Комментарий

Эти значения сохраняются для выборки с помощью функции **SPCSaveSample()**.

См. также

SPCSaveSample(), SPCSetRangeLimits(), SPCSetSpecLimits()

SPCSetMeasurement() SPC

Предоставляет удобный способ для ручного или автоматического ввода аналоговых измерения при выполнении сценария.

Синтаксис

```
SPCSetMeasurement("Dataset",Measurement,Value);
```

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
Measurement	Представляет номер замера (с 1 до 300) — константа или целый тэг.
Value	Значение, которое должно быть записано по указанному номеру замера. Константа или действительный тэг.

Комментарий Функция **SPCSaveSample()** используется, когда все замеры настроены на сохранение данных в базу.

SPCSetProductCollected() SPC

Изменяет имя продукта, для которого осуществляется сбор данных в указанном наборе данных.

Синтаксис

```
SPCSetProductCollected("Dataset","Product");
```

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
Product	Содержит имя продукта, под которым должны быть собраны данные. Фактическая строка или текстовый тэг.

Комментарий Эта функция не изменяет выведенный на дисплей продукт. Можно осуществлять сбор данных по одному продукту, а на дисплей выводить данные для другого продукта, используя эту функцию для сбора, а функцию **SPCProductDisplayed()** для вывода на дисплей.

Пример

```
SPCSetProductCollected("Data5838", "Widgets");
```

См. также **SPCSetProductDisplayed()**

SPCSetProductDisplayed() SPC

Заменяет продукт, выводимый на дисплей, в указанном наборе данных.

Синтаксис

```
SPCSetProductDisplayed("Dataset","Product");
```

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
Product	Содержит имя продукта, под которым данные должны быть выведены на дисплей. Фактическая строка или текстовый тэг.

Пример

```
SPCSetProductDisplayed("ADataSetName", "AProductName");
```

См. также **SPCSetProductCollected()**

SPCSetRangeLimits() SPC

Предоставляет удобный способ для ручного или автоматического ввода значений пределов управления для графа диапазонов.

Синтаксис **SPCSetRangeLimits**("*Dataset*",*RUCL*,*RLCL*);

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
RUCL	Представляет значение UCL графа диапазонов (верхний предел управления). Константа или действительный тэг.
RLCL	Представляет значение LCL графа диапазонов (нижний предел управления). Константа или действительный тэг.

См. также **SPCSetControlLimits(), SPCSetSpecLimits()**

SPCSetSpecLimits() SPC

Предоставляет удобный способ для ручного или автоматического ввода значений допустимых пределов характеристик для контрольного графа.

Синтаксис **SPCSetSpecLimits**("*Dataset*",*XUSL*,*XLSL*);

Параметр	Описание
Dataset	Содержит имя набора данных. Фактическая строка или текстовый тэг.
XUSL	Представляет значение USL графа (верхний допустимый предел). Константа или действительный тэг.
XLSL	Представляет значение LSL графа (нижний допустимый предел). Константа или действительный тэг.

См. также **SPCSetControlLimits(), SPCSetRangeLimits()**

SQLAppendStatement() SQL

Продолжает SQL-выражение, используя содержимое строки. При возникновении ошибок функция возвращает код ошибки.

Синтаксис [ResultCode=]**SQLAppendStatement**(*ConnectionID*,"*SQLStatement*");

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
SQLStatement	Actual statement to append.

Пример

```
[ResultCode=]SQLAppendStatement(ConnectionID,"where
tablename.columnname=(любое значение или строка)");
```

См. также SQLConnect()

SQLClearParam() SQL

Очищает значение указанного тэга. Теперь, прежде чем вызывать функцию **SQLExecute()**, нужно вызвать функцию **SQLSetParam()**.

Синтаксис

```
[ResultCode=]SQLClearParam(SQLHandle,ParameterNumber);
```

Параметр	Описание
SQLHandle	Целое число, возвращаемое SQL в результате выполнения функции **SQLPrepareStatement()**.
ParameterNumber	Целое число, возвращаемое SQL в результате выполнения функции SQLPrepareStatement().

См. также **SQLPrepareStatement(), SQLExecute()**

SQLClearStatement() SQL

Освобождает ресурсы, связанные с выражением, указанным аргументом *SQLHandle*.

Синтаксис

```
[ResultCode=]SQLClearStatement(ConnectionID,SQLHandle);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
SQLHandle	Целое число, возвращаемое SQL в результате выполнения функции **SQLPrepareStatement()**.

См. также **SQLConnect()**

SQLClearTable() SQL

Удаляет все записи в таблице базы данных, но сохраняет таблицу.

Синтаксис

```
[ResultCode=]SQLClearTable(ConnectionID,"TableName");
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой вы хотите получить доступ.

Пример Удалить все записи из таблицы BATCH1 можно следующим выражением:

```
[ResultCode=]SQLClearTable(ConnectionID,"BATCH1");
```

См. также **SQLConnect()**

SQLCommit() SQL

Команда **SQLCommit()** определяет конец группы команд транзакции. Группа команд, выполняемая между командой **SQLTransact()** и командой **SQLCommit()**, называется серией транзакций. Серия транзакций обрабатывается как одна транзакция. После того, как выдана команда **SQLTransact()**, все последующие операции не будут обращаться к базе данных, пока не будет выдана команда **SQLCommit()**.

Синтаксис

```
[ResultCode=]SQLCommit(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Примечание. Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример

```
ResultCode = SQLTransact( ConnectionID);
ResultCode = SQLInsertPrepare( ConnectionID, TableName,
BindList, SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertEnd( ConnectionID, SQLHandle );
ResultCode = SQLCommit( ConnectionID);

{Database keeps the 3 Inserts }
```

См. также SQLRollback(), SQLTransact()

SQLConnect() SQL

Соединяет систему InTouch с базой данных, указанной в строке *ConnectString*.

Синтаксис

```
[ResultCode=]SQLConnect(ConnectionID,"ConnectString");
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
ConnectString	Строка, определяющая базу данных и любую дополнительную информацию по загрузке, используемую в функции **SQLConnect()**.

Пример

Следующее выражение осуществляет соединение с СУБД IBM OS/2 Database Manager и базой данных по имени SAMPLE:

```
[ResultCode=]SQLConnect(ConnectionID,"DSN=OS2DM;DB=SAMPLE");
```

Эта функция возвращает значение в переменную *ConnectionID*, которая используется в качестве аргумента во всех функциях SQL.

Строка *ConnectString* задает систему управления базой данных и любую дополнительную информацию по загрузке. Она вводится в следующем формате:

```
"DSN=data source name[;attribute=value[;attribute=value]...]"
```

Для разных баз данных требуются различные атрибуты. QELIB распознает следующие атрибуты для всех СУБД:

📖 Подробная информация об атрибутах, поддерживаемых различными СУБД, содержится в главе 2 "*Руководства пользователя SQL Access для InTouch*".

Атрибут	Значение
DSN	Имя источника данных, сконфигурированное программой Microsoft ODBC Administrator.
DLG	Если разрешено (DLG=1), выводит на дисплей диалоговое окно, позволяющее вам вводить информацию в строке соединения.
DRV	Для совместимости с версией 4.11 SQL Access для InTouch. Это значение используется в имени источника данных (Data Source Name - DSN), но не присутствует в строке соединения. QELIB заменяет его на имя источника данных.
UID	Идентификационный номер для загрузки.
PWD	Пароль.
MODIFYSQL	Используется библиотекой QELIB для обеспечения совместимости между SQL, используемым в приложении, и SQL, используемым в базе данных. Если установлено в 1 (по умолчанию), драйвер базы данных ожидает ODBC-совместимый синтаксис, который она модифицирует, насколько необходимо, для специфической базы данных. Если установлено в 0, драйвер базы данных ожидает и поддерживает родной

	синтаксис специфической базы данных. Это позволяет вам продолжать использовать приложения, разработанные с SQL, поддерживаемым в версии 1.0 QELIB.
REREADAFTERUPDATE	Если разрешено (установлено в 1), QELIB повторно считывает запись из базы данных после ее обновления. Это полезно для получения корректного значения автоматически обновляемых столбцов, таких как метка времени.
REREADAFTERINSERT	Если разрешено (установлено в 1), QELIB повторно считывает запись из базы данных после вставки новых записей. Это полезно для получения корректного значения автоматически обновляемых колонок, таких как метка времени.

SQLCreateTable() SQL

Создает таблицу в базе данных, используя тэги в шаблоне таблицы с именем *TemplateName*. Шаблоны таблицы (задаваемые в файле SQL.DEF) определяют структуру таблицы базы данных.

Синтаксис

```
[ResultCode=]SQLCreateTable(ConnectionID,TableName,TemplateName);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.
TemplateName	Имя определения шаблона, который нужно использовать.

Пример

Это выражение создает таблицу с именем *BATCH1* с именами и типами полей (столбцов), определенными в шаблоне *TEMPLATE*:

```
[ResultCode=]SQLCreateTable(ConnectionID,"BATCH1","TEMPLATE");
```

Примечание. Если параметр вводится в сценарий в окружении двойных кавычек, например «Parametre1», тогда используется в точности эта строка. Если не стоит никаких кавычек, то в этом случае считается, что Parameter1 — это тэг, и система будет обращаться к словарю InTouch за значением параметра. Например:

```
"c:\main\file" в отличие от location
```

где: location — текстовый тэг InTouch

```
"c:\main\file" — буквальная текстовая строка
```

См. также **SQLConnect()**

SQLDelete() SQL

Удаляет запись или несколько записей.

Синтаксис

```
[ResultCode=]SQLDelete(ConnectionID,TableName,WhereExpr);
```

Примечание. Функции **SQLDelete()** нельзя задавать нулевое выражение *WhereExpression*.

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.

WhereExpression	Определяет условие, которое может быть либо Истиной, либо Ложью для любой строки в таблице. Функция выбирает из таблицы только те строки, для которых это условие истинно. Это выражение должно быть в следующем формате:
	ИмяСтолбца *оператор_сравнения* выражение
	Примечание. Если столбец содержит данные символьного типа, выражение должно быть в одинарных кавычках.
	В приводимом примере будут выбраны все строки, чей столбец имен содержит значение EmployeeID:
	`name='EmployeeID'`
	В следующем примере будут выбраны все строки, содержащие в поле partno номера от 100 до 199:
	`partno>=100 and partno<200`
	В следующем примере будут выбраны все строки, содержащие в поле temperature значения больше 350:
	`temperature>350`

Пример

Следующее выражение удаляет из таблицы *BATCH1* все записи, в которых номер партии (lot number) равен 65:

```
[ResultCode=]SQLDelete(ConnectionID,"BATCH1","lotno=65");
```

См. также **SQLConnect()**

SQLDisconnect() SQL

Отсоединяет пользователя от базы данных.

Синтаксис

```
[ResultCode=]SQLDisconnect(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

См. также SQLConnect()

SQLDropTable() SQL

Разрушает таблицу.

Синтаксис `[ResultCode=]SQLDropTable(ConnectionID,TableName);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.

Пример Это выражение разрушает таблицу *BATCH1*. После того, как будет дана такая команда, имя таблицы больше не будет распознаваться и не будет отклика на любые другие команды:

```
[ResultCode=]SQLDropTable(ConnectionID,"BATCH1");
```

См. также **SQLConnect()**

SQLEnd() SQL

Эта функция применяется после функции **SQLSelect()** для освобождения ресурсов, которые использовались для хранения таблицы результатов.

Синтаксис `[ResultCode=]SQLEnd(ConnectionID);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если не установлено — отрицательное целое.

См. также **SQLConnect(), SQLSelect()**

SQLErrorMsg() SQL

Получает текстовое сообщение об ошибке, связанное с указанным кодом результата *ResultCode*. *ErrorMsg* является внутренним тэгом типа сообщение (максимум 131 символ), связанным с *ResultCode*.

Синтаксис `SQLErrorMsg(ResultCode);`

Параметр	Описание
ResultCode	Целая переменная, возвращаемая почти всеми SQL функциями. Она равна нулю, если функция завершилась успешно, и отрицательному целому, если был сбой.

☞ Подробное описание кодов ошибок дано в Приложении A, "Отладка функций сценариев SQL".

Пример

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

См. также **SQLConnect()**

SQLExecute() SQL

Выполняет выражение SQL. Если выражение является выражением выбора, тэг BindList обозначает имя списка связей (Bind List) для связывания столбцов базы данных с тэгами InTouch. Если список связей нулевой, не будет сформировано никаких отношений тэгов. Например, выражением SQL могут быть Create (Создать), View (Просмотреть), Insert (Вставить) и т. д. Функция возвращает код ошибки. Если выражение было «подготовлено», то в функцию **SQLExecute** в качестве аргумента передается идентификатор выражения (результат функции **SQLPrepareStatement**). Если выражение не было «подготовлено», идентификатор выражения будет равен 0.

Синтаксис

```
[ResultCode=]SQLExecute(ConnectionID,BindList,SQLHandle);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если не установлено — отрицательное целое.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.

См. также **SQLConnect(),SQLPrepareStatement()**

Примечание. Функция **SQLExecute()** может быть вызвана только один раз для выражения, которое не было «подготовлено». Если выражение было «подготовлено», эта функция может вызываться много раз.

SQLFirst() SQL

Выбирает первую запись из таблицы результатов, созданной последним вызовом функции **SQLSelect()**. Функция **SQLSelect()** должна быть обработана перед использованием этой команды.

Синтаксис

```
[ResultCode=]SQLFirst(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

См. также **SQLConnect(), SQLSelect()**

SQLGetRecord() SQL

Считывает запись, заданную номером записи *Record Number*, из буфера текущего выбора.

Синтаксис

```
[ResultCode=]SQLGetRecord(ConnectionID,Record Number);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
RecordNumber	Фактический номер извлекаемой записи.

Пример `[ResultCode=]SQLGetRecord(ConnectionID,3);`

См. также **SQLConnect()**

SQLInsert() SQL

Вставляет новую запись в указанную таблицу, используя значения тэгов в прилагаемом списке связей (BindList).

Синтаксис `[ResultCode=]SQLInsert(ConnectionID,TableName,BindList);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.

Пример Следующее выражение вставляет новую запись в таблицу ORG со значениями тэгов, заданными в List1:

```
[ResultCode=]SQLInsert(ConnectionID,"ORG","List1");
```

Примечание. Следующие три функции можно использовать вместо стандартной функции **SQLInsert()** для быстрой вставки записей в файл. Функция **SQLInsert()** это одношаговая операция, которая делает вставку и завершает выполнение выражения. Поэтому, в следующий раз, когда будет вызвана эта функция, вся операция будет проделана еще раз. Это займет гораздо больше времени, чем использование трех приведенных ниже функций. Эти три функции разбивают на части выполнение шагов операции, так, что если вы один раз выполните функцию **SQLInsertPrepare()**, вы можете затем вызывать функцию **SQLInsertExecute()** сколько угодно раз, довольно быстро, а затем для завершения вызвать функцию **SQLInsertEnd()**.

См. также **SQLConnect(), SQLInsert(), SQLInsertPrepare(), SQLInsertExecute(), SQLInsertEnd()**

SQLInsertEnd() SQL

Освобождает выражение.

Синтаксис `[ResultCode=]SQLInsertEnd(ConnectionID,SQLHandle);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено

	успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.

См. также **SQLConnect(), SQLPrepareStatement()**

SQLInsertExecute() SQL

Выполняет заранее подготовленное выражение.

Синтаксис

```
[ResultCode=]SQLInsertExecute(ConnectionID,BindList,SQLHandle);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.

См. также **SQLConnect(), SQLPrepareStatement()**

SQLInsertPrepare() SQL

Создает и подготавливает выражение вставки для выполнения. Выражение Insert (вставки) не обрабатывается. В результате выполнения *InsertPrepare()* тэг целого типа *SQLHandle* будет содержать указатель SQL выражения.

Синтаксис

```
[ResultCode=]SQLInsertPrepare(ConnectionID,TableName,BindList,
                              SQLHandle);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.

См. также **SQLConnect(), SQLPrepareStatement()**

SQLLast() SQL

Осуществляет выбор последней записи в Таблице Результатов, созданной последним вызовом функции **SQLSelect()**. Функция **SQLSelect()** должна быть выполнена до использования этой команды.

Синтаксис

```
[ResultCode=]SQLLast(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример `[ResultCode=]SQLLast(ConnectionID);`

См. также **SQLConnect(), SQLSelect()**

SQLLoadStatement() SQL

Считывает выражение, содержащееся в файле *filename*. Это выражение похоже на создаваемое функцией **SQLSetStatement()** и может быть добавлено с помощью функции **SQLAppendStatement()** или обработано функцией **SQLExecute()**. В файле может быть только одно выражение. Однако, функцию **SQLAppendStatement()** можно применить для добавления чего-либо к выражению, если функция **SQLPrepareStatement()** или **SQLExecute()** еще не вызывались.

Синтаксис `[ResultCode=]SQLLoadStatement(ConnectionID,FileName);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
FileName	Имя файла, где содержится информация.

Комментарий Подготовьте SQL созданное выражение либо функцией **SQLSetStatement()**, либо **SQLLoadStatement()**.

Пример

[ResultCode=]SQLLoadStatement**(ConnectionID,"C:\InTouchAppname\SQL.txt")**

```
SQL.txt = Select ColumnName from TableName where
ColumnName>100;
```

См. также **SQLConnect(), SQLAppendStatement(), SQLExecute(), SQLPrepareStatement**

SQLManageDSN() SQL

Запускает программу конфигурирования драйверов ODBC Microsoft ODBC Manager. Ее можно использовать для добавления, удаления и модификации всех имен источников данных (data source names).

Синтаксис `SQLManageDSN(ConnectionID);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

SQLNext() SQL

Осуществляет выбор следующейей записи в Таблице Результатов, созданной последним вызовом функции **SQLSelect()**. Функция **SQLSelect()** должна быть выполнена до использования этой команды.

Синтаксис

```
[ResultCode=]SQLNext(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример

```
[ResultCode=]SQLNext(ConnectionID);
```

См. также **SQLConnect(), SQLSelect()**

SQLNumRows() SQL

Показывает, сколько строк в таблице удовлетворяет критерию, указанному в последнем вызове функции **SQLSelect()**. Например, если условиие *WhereExpression* используется для выбора всех строк с именем столбца AGE, где AGE равно 45, возвращаемое количество строк может быть 40 или 4000. По этому результату можно решить, какую следующую функцию надо выполнить.

Синтаксис

```
SQLNumRows(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример Следующее выражение возвращает число выбранных строк в значение целого тэга NumRows:

```
NumRows=SQLNumRows(ConnectionID);
```

См. также **SQLConnect()**

SQLPrepareStatement() SQL

Функция **SQLSelect()** должна быть выполнена до вызова этой команды. Подготовьте выражение SQL, которое было создано либо функцией **SQLSetStatement()**, либо функцией **SQLLoadStatement()**. Возвращается идентификатор выражения.

Синтаксис

```
[ResultCode=]SQLPrepareStatement(ConnectionID,SQLHandle);
```

Параметр	Описание

ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.

Пример `[ResultCode=]SQLPrepareStatement(ConnectionID,SQLHandle);`

См. также **SQLConnect(), SQLSelect(), SQLSetStatement(), SQLLoadStatement()**

SQLPrev() SQL

Осуществляет выбор предыдущей записи в Таблице Результатов, созданной последним вызовом функции **SQLSelect()**.

Синтаксис `[ResultCode=]SQLPrev(ConnectionID);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Комментарий Перед использованием этой команды должна быть выполнена функция **SQLSelect()**.

Пример
```
[ResultCode=]SQLPrev(ConnectionID);
```

См. также **SQLConnect(), SQLSelect()**

SQLRollback() SQL

Команда **SQLRollback()** возвращает, или «откатывает назад» последнюю выполненную серию транзакций. Группа команд, выполняемая между командой **SQLTransact()** и командой **SQLCommit()**, называется серией транзакций. Серия транзакций обрабатывается как одна транзакция. После того, как выдана команда **SQLTransact()**, все последующие операции не будут обращаться к базе данных, пока не будет выдана команда **SQLCommit()**.

Синтаксис `[ResultCode=]SQLRollback(ConnectionID,);`

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции SQLConnect(). Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример
```
ResultCode =SQLTransact( ConnectionID);
ResultCode = SQLInsertPrepare( ConnectionID, TableName,
BindList, SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertExecute( ConnectionID, BindList,
SQLHandle );
ResultCode = SQLInsertEnd( ConnectionID, SQLHandle );
ResultCode =SQLRollback( ConnectionID);

{Ignores all commands after Transact()  Database unchanged }
```

См. также **SQLCommit(), SQLTransact()**

SQLSelect() SQL

Дает команду базе данных выбрать информацию из таблиц. При выполнении **SQLSelect()** в памяти создается временная таблица результатов, содержащая записи, которые можно просмотреть функциями **SQLFirst()**, **SQLLast()**, **SQLNext()** и **SQLPrev()**.

Примечание. Нужно всегда использовать функцию **SQLEnd()** после **SQLSelect()** для освобождения ресурсов, которые были заняты для таблицы результатов.

Синтаксис

```
[ResultCode=]SQLSelect(ConnectionID,TableName,BindList,
WhereExpr,OrderByExpr);
```

Параметры	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой требуется доступ.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.

WhereExpression	Определяет условие, которое может быть либо Истиной, либо Ложью для любой строки в таблице. Функция выбирает из таблицы только те строки, для которых это условие истинно. Это выражение должно быть в следующем формате:
	ИмяСтолбца *оператор_сравнения* выражение
	Примечание. Если столбец содержит данные символьного типа, выражение должно быть в одинарных кавычках.
	В приводимом примере будут выбраны все строки, чей столбец имен содержит значение EmployeeID:
	`name='EmployeeID'`
	В следующем примере будут выбраны все строки, содержащие в поле partno номера от 100 до 199:
	`partno>=100 and partno<200`
	В следующем примере будут выбраны все строки, содержащие в поле temperature значения больше 350:
	`temperature>350`

Примеры WhereExpression:

WhereExpr – внутренний текстовый тэг

OrderByExpr - внутренний текстовый тэг

Speed_Input – внутренний действительный – аналоговый ввод пользователя

Serial_Input – внутренний текстовый – строка ввода пользователя

Аналоговый пример

```
WhereExpr = «Speed = « + text(Speed_Input,»#.##»);
```

Поскольку *Speed_Input* является числом, его необходимо преобразовать в текст, который можно конкатенироватьк строке выражения *where*.

Строковый пример

```
WhereExpr = «Ser_No = ‘« + Serial_input + «’»;
```

- Поскольку *Serial_Input* является строковым значением, оно должно быть заключено в ординарные кавычки: WhereExpr = «Ser_No=’125gh’»;

Строковый пример с конструкцией like

```
WhereExpr = «Ser_No like ‘» + «125%’»
```

- С выражением Like можно использовать символ % в качестве символа подстановки.

Строковый и аналоговый пример с конструкцией And

```
WhereExpr = «Ser_No = ‘« + Serial_input + «’» + « and « +
«Speed = « + text(Speed_Input,»#.##»);
OrderByExpr = «»;
```

- Если порядок сортировки не имеет значения, можно использовать нулевую строку, как показано выше.

SQLSelect с тэгом WhereExpr

```
ResultCode =
SQLSelect(Connect_Id,TableName,BindList,WhereExpr,OrderByExpr
);

Error_msg = SQLErrorMsg( ResultCode );
```

Встроенная функция SQLSelect WhereExpr

```
ResultCode =    SQLSelect(Connect_Id,TableName,BindList,
«Ser_No = ‘« + Serial_input + «’», OrderByExpr);

Error_msg = SQLErrorMsg( ResultCode );
```

Примечание. После выполнения функции **SQLSelect()** необходимо всегда вызывать команду **SQLEnd**(*Connect_Id*). Иначе не будут освобождаться ресурсы, и приложение приведет к нехватке памяти.

OrderByExpression	Определяет столбец и направление сортировки. Только имя столбца используется для сортировки, и выражение должно быть задано в такой форме: ColumnName [ASC|DESC] В данном примере в выбранной таблице по столбцу «manager» осуществляется сортировка в возрастающем порядке: "manager ASC" Вы можете также сортировать по нескольким столбцам, задавая выражение в следующей форме: ColumnName [ASC|DESC], ColumnName [ASC|DESC] В следующем примере осуществляется сортировка по столбцу температуры в возрастающем порядке и по столбцу времени в убывающем порядке: "temperature ASC, time DESC"

Пример

В следующем выражении выбираются записи из таблицы BATCH, используя Список Связей по имени List1, у которых в столбце под названием «type» содержится значение «cookie». Информация будет представлена отсортированной по столбцу «amount» в возрастающем порядке и по столбцу «sugar» в убывающем порядке:

```
[ResultCode=]SQLSelect(ConnectionID,"BATCH","List1","type='co
okie'"
,"amount ASC, sugar DESC");
```

А это выражение выбирает все данные в базе данных, не указывая значение для оператора сравнения и порядка сортировки:

```
[ResultCode=]SQLSelect(ConnectionID,"BATCH","List1","","");
```

См. также SQLFirst(), SQLConnect(), SQLLast(), SQLNext(), SQLPrev(), SQLEnd(), SQLSelect()

SQLSetParamChar() SQL

Устанавливает значение указанного тэга равным значению указанной строки. Функция **SQLSetParamChar()** может быть вызвана многократно, в результате будет осуществлена конкатенация (последовательное соединение) всех значений. Длина 0 игнорируется.

Синтаксис

```
[ResultCode=]SQLSetParamChar(SQLHandle,ParameterNumber,
ParameterValue,MaxLen);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое надо установить.
MaxLen	Максимальный размер столбца, с которым связан данный тэг. Эта установка определяет, относится тэг к переменному символьному типу или к длинному переменному символьному типу. Если *MaxLen* меньше или равно наибольшей длине, допустимой в базе данных, то этот тэг относится к переменному символьному типу. Если больше — то к длинному переменному символьному типу.

Пример

```
[ResultCode=]SQLSetParamChar(SQLHandle,ParameterNumber,
ParameterValue,MaxLen);
```

См. также **SQLPrepareStatement()**

SQLSetParamDate() SQL

Задает указанному тэгу даты значение указанной строки.

Синтаксис

```
[ResultCode=]SQLSetParamDate(SQLHandle,ParameterNumber,
ParameterValue);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция SQLPrepareStatement().
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.

Пример

```
[ResultCode=]SQLSetParamDate(SQLHandle,ParameterNumber,
ParameterValue);
```

См. также **SQLPrepareStatement()**

SQLSetParamDateTime() SQL

Задает указанному тэгу даты и времени значение указанной строки.

Синтаксис

```
[ResultCode=]SQLSetParamDateTime(SQLHandle,ParameterNumber,
ParameterValue,Precision);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.
Precision	Длина присваемого значения даты-времени. Это количество символов в значении *ParameterValue*.

Пример

```
[ResultCode=]SQLSetParamDateTime(SQLHandle,ParameterNumber,
ParameterValue,Precision);
```

См. также **SQLPrepareStatement()**

SQLSetParamDecimal() SQL

Присваивает заданное значение *ParameterValue* указанному десятичному тэгу. *Precision* — это количество цифр (знаков) в значении, а *Scale* — количество знаков справа от десятичной точки.

Синтаксис

```
[ResultCode=]SQLSetParamDecimal(SQLHandle,ParameterNumber,
ParameterValue,Precision,Scale);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.
Precision	Длина (количество знаков) присваемого десятичного значения.
Scale	Количество знаков справа от десятичной точки.

Пример

```
[ResultCode=]SQLSetParamDecimal(SQLHandle,ParameterNumber,
ParameterValue,Precision, Scale);
```

См. также **SQLPrepareStatement()**

SQLSetParamFloat() SQL

Присваивает заданное значение *ParameterValue* указанному тэгу.

Синтаксис

```
[ResultCode=]SQLSetParamFloat(SQLHandle,ParameterNumber,
ParameterValue);
```

Параметр	Описание
SQLHandle	Целое значение, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.

Пример

```
[ResultCode=]SQLSetParamFloat(SQLHandle,ParameterNumber,
ParameterValue);
```

См. также **SQLPrepareStatement()**

SQLSetParamInt() SQL

Присваивает заданное значение *ParameterValue* указанному тэгу.

Синтаксис

```
[ResultCode=]SQLSetParamInt(SQLHandle,ParameterNumber,
ParameterValue);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.

Пример

```
[ResultCode=]SQLSetParamInt(SQLHandle,ParameterNumber,
ParameterValue);
```

См. также **SQLPrepareStatement()**

SQLSetParamLong() SQL

Присваивает заданное значение *ParameterValue* указанному тэгу.

Синтаксис

```
[ResultCode=]SQLSetParamLong(SQLHandle,ParameterNumber,
ParameterValue);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.

Пример

```
[ResultCode=]SQLSetParamLong(SQLHandle,ParameterNumber,
ParameterValue);
```

См. также **SQLPrepareStatement()**

SQLSetParamNull() SQL

Присваивает значение NULL указанному тэгу.

Синтаксис

```
[ResultCode=]SQLSetParamNull(SQLHandle,ParameterNumber,
ParameterType,Precision,Scale);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция **SQLPrepareStatement()**.
ParameterNumber	Фактический номер тэга в выражении.
ParameterType	Тип данных указанного тэга:

Тип	Значение	Описание
Char	1	Строка фиксированной длины, забитая пробелами
Var Char	2	Строка переменной длины
Decimal	3	Число BCD (десятичное число в двоичной кодировке)
Integer	4	4-байтовое знаковое целое
Small integer	5	2-байтовое знаковое целое
Float	6	4-байтовое число с плавающей точкой
Double Precision Float	7	8-байтовое число с плавающей точкой
DateTime	8	26-байтовое значение даты и времени
Date	111	26-байтовое значение даты
Time	112	26-байтовое значение времени
No Type	0	Нет типа данных

Precision	Длина десятичного значения, максимальная длина символьной строки или длина в байтах значения даты-времени.
Scale	Количество знаков справа от десятичной точки. Это значение требуется только, если оно применимо к тэгу, который устанавливается в ноль.

Комментарий Можно не указывать тип тэга, если для этого тэга была уже вызвана функция **SQLSetParam()**;

Пример

```
[ResultCode=]SQLSetParamNull(SQLHandle,ParameterNumber,
ParameterType,Precision,Scale);
```

См. также **SQLPrepareStatement()**

SQLSetParamTime() SQL

Устанавливает значение указанного тэга времени равным значению указанной строки.

Синтаксис

```
[ResultCode=]SQLSetParamTime(SQLHandle,ParameterNumber,
ParameterValue);
```

Параметр	Описание
SQLHandle	Целое значение, возвращаемое SQL, если используется функция SQLPrepareStatement().
ParameterNumber	Фактический номер тэга в выражении.
ParameterValue	Фактическое значение, которое нужно установить.

Пример

```
[ResultCode=]SQLSetParamTime(SQLHandle,ParameterNumber,
ParameterValue);
```

См. также **SQLPrepareStatement()**

SQLSetStatement() SQL

Установленному идентификатору соединения *ConnectionID* ставится в соответствие буфер, используя содержимое заданной строки *String*. Может быть только один буфер SQL выражения на один идентификатор соединения *ConnectionID*. В результате выполнения функция возвращает код ошибки.

Синтаксис

```
[ResultCode=]SQLSetStatement(ConnectionID,SQLStatement);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
SQLStatement	Фактическое выражение, см. ниже пример.

Пример

```
[ResultCode=]SQLSetStatement(ConnectionID,"Select LotNo,
LotName
from LotInfo");
```

В следующем выражении SQLHandle устанавливается в 0, чтобы не нужно было вызывать SQLPrepare(Connect_Id, SQLHandle) перед выполнением выражения. Поскольку SQLhandle не был создан функцией SQLPrepare, то для правильного завершения выбора следует использовать функцию SQLEnd вместо SQLClearStatement().

```
SQLSetStatement( Connect_Id, «Select  Speed, Ser_No from
tablename where Ser_No =’» +   Serial_input  + «’»);

SQLExceute(Connect_Id,0);
```

В следующем выражении SQLhandle создается функцией SQLPrepareStatement и используется в SQLExecute. Для завершения выбора следует использовать функцию SQLClearStatement, чтобы освободить ресурсы и освободить SQLhandle.

```
SQLSetStatement( Connect_Id, «Select  Speed, Ser_No from
tablename where Ser_No =’» +   Serial_input  + «’»);
SQLPrepareStatement(Connect_Id,SQlHandle);
```

```
SQLExceute(Connect_Id,SqlHandle);
```

См. также SQLConnect()

SQLTransact() SQL

Команда **SQLTransact()** определяет начало группы команд транзакции. Группа команд, выполняемая между командой **SQLTransact()** и командой **SQLCommit()**, называется серией транзакций. Серия транзакций обрабатывается как одна транзакция. После того, как выдана команда **SQLTransact()**, все последующие операции не будут обращаться к базе данных, пока не будет выдана команда **SQLCommit()**.

Синтаксис

```
[ResultCode=]SQLTransact(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Примечание. Будьте внимательны при написании сценария, включающего команду **SQLCommit()**. Поскольку время обработки умножается на количество команд в серии транзакций, включение многочисленных команд значительно замедлит выполнение.

См. также **SQLCommit(), SQLRollback()**

SQLUpdate() SQL

Модифицирует запись в базе данных, обновляя ее в соответствии с текущим значением тэгов, указанных в Списке Связей (Bind List).

Синтаксис

```
[ResultCode=]SQLUpdate(ConnectionID,TableName,BindList,
WhereExpr);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.
TableName	Имя таблицы базы данных, к которой нужен доступ.
BindList	Определяет, какие тэги InTouch используются и с какими столбцами базы данных они связаны.
WhereExpression	Определяет условие, которое может быть либо Истиной, либо Ложью для любой строки в таблице. Функция выбирает из таблицы только те строки, для которых это условие истинно. Это выражение должно быть в следующем формате: ИмяСтолбца *оператор_сравнения* выражение

Пример

В следующем выражении обновляются записи из таблицы BATCH, чей номер партии равен 65, в соответствии с текущими значениями тэгов, указанных в списке связей под названием "List1":

```
[ResultCode=]SQLUpdate(ConnectionID,"BATCH","List1","lotno=65");
```

Примечание. Убедитесь в том, что все записи являются уникальными. Если в таблице существуют одинаковые записи, обе будут обновлены.

См. также **SQLConnect()**

SQLUpdateCurrent() SQL

Берет текущую выбранную запись в базе данных и обновляет ее в соответствии с текущими значениями InTouch. В примере обновляется текущая выбранная запись.

Синтаксис

```
[ResultCode=]SQLUpdateCurrent(ConnectionID);
```

Параметр	Описание
ConnectionID	Это значение возвращается при успешном выполнении функции **SQLConnect()**. Если соединение установлено успешно, это будет положительное целое. Если соединение не установлено, это будет отрицательное целое.

Пример

```
[ResultCode=]SQLUpdateCurrent(ConnectionID);
```

См. также **SQLConnect()**

Sqrt() математическая

Заставляет систему InTouch автоматически вычислять квадратный корень заданного числа.

Синтаксис

```
Real Result=Sqrt(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.
Real Result	Тэг действительного типа, в котором будет храниться результат.

Пример

```
AnalogTag1=Sqrt(AnalogTag2);
```

StartApp системная

Автоматически запускает другое приложение Windows.

Синтаксис

```
StartApp "AppName";
```

Параметр	Описание
AppName	Фактическое имя программы приложения, которое надо запустить, например *Wordpad.exe*.

Рекомендуется указывать расширение программы .EXE. Можно также добавить в командную строку ключи программы, если они поддерживаются.

Длинные имена файлов не поддерживаются, однако можно использовать следующие Dos-эквиваленты длинных имен.

Если длинное имя пути файла — C:\Program files\Microsoft Office\Office\Excel. Используйте C:\Progra~1\Micros~2\Office\Excel (Dos-эквивалент).

```
SetApp "C:\Progra~1\Micros~2\Office\Excel";
```

При использовании файла, созданного в среде Windows, отображаются все свойства файла, а в центральном столбце указывается Dos-эквивалент пути файла.

Примеры Следующее выражение запускает программу Microsoft Windows Wordpad.

```
StartApp "Wordpad.exe";
```

См. также **ActivateApp()**

StringASCII() строка

Возвращает ASCII значение первого символа в указанном тэге типа сообщение.

Синтаксис

```
IntegerResult=StringASCII("Char");
```

Параметр	Описание
Char	Буква или цифра, или текстовый тэг.

Комментарий ASCII значение первого символа, заданного аргументом Char, будет возвращено в тэг IntegerResult. При выполнении функция обрабатывает только один символ. Если в текстовом тэге дана строка ASCII, состоящая из более, чем из одного символа, будет проанализирован только первый символ строки.

Примеры

```
StringASCII("A") возвращает 65
StringASCII("A Mixer is Running") возвращает 65
StringASCII("a mixer is running") возвращает 97
```

См. также **StringChar()**

StringChar() строка

Возвращает символ, соответствующий указанному ASCII коду.

Синтаксис

```
MessageResult=StringChar(ASCII);
```

Параметр	Описание
ASCII	Код ASCII или целый тэг.

Комментарий ASCII символ, заданный аргументом *ASCII*, будет возвращен в тэг *MessageResult*. Одним из примеров применения этой функции может быть добавление в строку текстового тэга таких ASCII символов, которые нельзя ввести с клавиатуры.

Пример

```
ControlString=MessageTag+StringChar(13)+StringChar(10);
```

Коды [CR] (возврат каретки) и [LF] (перевод строки) добавлены в конец строки *MessageTag*, и результат записан в *ControlString*. Вставка в строку невыводимых символов (с кодами вне диапазона 32-126), может быть весьма полезной для создания управляющих кодов, предназначенных для внешних устройств, таких как принтеры или модемы.

Эта функция обычно применяется с командами SQL. Конструкция "where" часто требует двойных кавычек вокруг строковых переменных, поэтому используется **StringChar**(34).

См. также **StringASCII()**

StringFromIntg() строка

Преобразует целое значение в строковое представление по другому основанию.

Синтаксис

```
MessageResult=StringFromIntg(Number,Base);
```

Параметр	Описание
Number	Преобразуемое число — константа или целый тэг.
Base	Основание для преобразования — константа или целый тэг.

Комментарий Осуществляется преобразование числа *Integer* по указанному основанию *Base*, и результат записывается в *MessageResult*.

Примеры

```
StringFromIntg(26, 2) возвращает "11010"
StringFromIntg(26, 8) возвращает "32"
StringFromIntg(26, 16) возвращает "1A"
```

См. также **StringToIntg(), StringFromReal(), StringToReal(), Text()**

StringFromReal() строка

Преобразует в строковое представление действительное значение, выводимое либо как число с плавающей точкой, либо в экспоненциальном виде.

Синтаксис

```
MessageResult=StringFromReal(Number,Precision,"Type");
```

Параметр	Описание
Number	Преобразуемое число. Преобразуется с заданной точностью (*Precision*) и типом (*Type*), результат заносится в MessageResult. Константа или действительный тэг.
Precision	Задает количество знаков после запятой, которые будут выведены в строке — константа или целый тэг.
Type	Определяет способ вывода:

Тип	Описание
"f"	Вывод в виде числа с плавающей точкой.
"e"	Вывод в экспоненциальном виде с буквой «e» в нижнем регистре.
"E"	Вывод в экспоненциальном виде с буквой «E» в верхнем регистре.

Примеры

```
StringFromReal(263.355, 2,"f") возвращает "263.36"
StringFromReal(263.355, 2,"e") возвращает "2.63e2"
StringFromReal(263.55, 3,"E") возвращает "2.636E2"
```

См. также **StringToReal(), StringFromIntg(), StringToIntg(), Text()**

StringFromTime() строка

Преобразует значение времени (в секундах с 1 января 1970) в строковое представление заданного формата.

Синтаксис

```
MessageResult=StringFromTime(SecsSince1-1-70,StringType);
```

Параметр	Описание
SecsSince1-1-70	Преобразуется в строку типа *StringType*, и результат заносится в *MessageResult*.
StringType	Определяет способ вывода:

Тип	Описание
1	Выводит дату в формате, заданном в панели управления Windows. (Аналогично формату $DateString).
2	Выводит время в формате, заданном в панели управления Windows. (Аналогично формату $TimeString).
3	Выводит 24-символьную строку, показывающую и дату и время: "Wed Jan 02 02:03:55 1993"
4	Выводит сокращенное название для недели: "Wed"
5	Выводит полное название для недели: "Wednesday"

Примеры

```
StringFromTime(86400, 1) возвращает "1/2/70"
StringFromTime(86400, 2) возвращает "12:00:00 AM"
StringFromTime(86400, 3) возвращает "Fri Jan 02 00:00:00 1970"
StringFromTime(86400, 4) возвращает "Fri"
StringFromTime(86400, 5) возвращает "Friday"
```

StringInString() строка

Возвращает позицию в тексте *Text*, где впервые встречается подстрока поиска *SearchFor*.

Синтаксис

```
IntegerResult=StringInString("Text","SearchFor",StartPos,
CaseSens);
```

Параметр	Описание
Text	Строка, в которой ищется вхождение строки *SearchFor*. Если обнаруживается многократное вхождение строки *SearchFor*, возвращается позиция первого вхождения и записывается в *IntegerTag*. Фактическая строка или текстовый тэг.
SearchFor	Искомая строка. Фактическая строка или текстовый тэг.
StartPos	Целое число, определяющее, с какой позиции исходной строки *Text* начнется поиск.
CaseSens	Тэг, определяющий будет ли поиск зависеть от регистра, в котором введена подстрока (0=нет, 1=да)— константа или целый тэг.

Пример

```
StringInString("The mixer is running", "mix", 1, 0)
возвращает 5

StringInString("Today is Thursday", "day", 1, 0) возвращает 3

StringInString("Today is Thursday", "day", 10, 0) возвращает
15

StringInString("Today is Veteran's Day", "Day", 1, 1) will
return 20

StringInString("Today is Veteran's Day", "Night", 1, 1) will
return 0
```

StringLeft() строка

Возвращает первый (крайний левый) символ(ы) в указанном тэге-сообщении.

Синтаксис

```
MessageResult=StringLeft("Text",Chars);
```

Параметр	Описание
Text	Фактическая строка или текстовый тэг
Chars	Количество возвращаемых символов.

Комментарий Если значение *Chars* установлено в 0, будет возвращена вся строка.

Пример

```
StringLeft("The Control Pump is On", 3) возвращает "The"

StringLeft("Pump 01 is On", 4) возвращает "Pump"

StringLeft("Pump 01 is On", 96) возвращает "Pump 01 is On"

StringLeft("The Control Pump is On", 0) возвращает "The
Control

Pump is On"
```

StringLen() строка

Возвращает целое значение длины указанного текстового тэга.

Синтаксис

```
IntegerResult=StringLen("Text");
```

Параметр	Описание
Text	Фактическая текстовая строка или текстовый тэг.

Комментарий Длина (в символах) текстовой строки *Text* возвращается в *IntegerTag*. Подсчитываются все символы в сообщении, включая и те, что не выводятся на экран.

Пример

```
StringLen("Twelve percent") возвращает 14
StringLen("12%") возвращает 3
StringLen("The end." + StringChar(13)) возвращает 10
```

Примечание. Символ возврата каретки [CR] имеет ASCII код 13.

StringLower() строка

Преобразует все символы в верхнем регистре в нижний регистр в заданном текстовом тэге.

Синтаксис

```
MessageResult=StringLower("Text");
```

Параметр	Описание
Text	Фактическая строка или текстовый тэг.

Комментарий Преобразование не повлияет на символы нижнего регистра, цифры, знаки и друие специальные символы.

Примеры

```
StringLower("TURBINE") возвращает "turbine"
StringLower("22.2 Is The Value") возвращает "22.2 is the value."
```

StringMid() строка

Возвращает заданное число символов из указанного текстового тэга, начиная с указанной позиции. Эта функция немного отличается от функций этой группы **StringLeft()** и **StringRight()** тем, что она позволяет пользователю указать обе начальную и конечную позицию строки, из которой должна быть извлечена подстрока.

Синтаксис

```
MessageResult=StringMid("Text",StartChar,Chars);
```

Параметр	Описание
Text	Фактическая строка или текстовый тэг.
StartChar	Указывает позицию первого символа извлекаемой подстроки — константа или целый тэг.
Chars	Количество возвращаемых символов — константа или целый тэг.

Примеры

```
StringMid("The Furnace is Overheating",5,7,) возвращает
 "Furnace"
StringMid("The Furnace is Overheating",13,3) возвращает "is "
StringMid("The Furnace is Overheating",16,50) возвращает
"Overheating"
```

См. также **StringLeft(), StringRight()**

StringReplace() строка

Заменяет или изменяет указанную часть заданной строки. С помощью этой функции можно брать текстовый тэг и заменять символы, слова или фразы.

Синтаксис

```
MessageResult=StringReplace(Text,SearchFor,ReplaceWith,
CaseSens,NumToReplace,MatchWholeWords);
```

Параметр	Описание
Text	Изменяемый текст. Фактическая строка или текстовый тэг.
SearchFor	Строка, которую нужно найти и заменить в ней текст. Фактическая строка или текстовый тэг.
ReplaceWith	Новая строка, заменяющая старую. Фактическая строка или текстовый тэг.
CaseSens	Определяет, будет ли поиск зависеть от регистра (0=нет, 1=да). Константа или целый тэг.
NumToReplace	Определяет число вхождений, которое нужно заменить (-1 = все). Константа или целый тэг.
MatchWholeWords	Определяет, будет ли функция заменять только полные слова (0=нет, 1=да). Константа или целый тэг. Если значение *MatchWholeWords* равно 1, а значение *SearchFor* равно «and», то в слове «handle» подстрока «and» не будет заменена. Если же значение *MatchWholeWords* будет равно 0, такая замена будет осуществлена.

Примеры

```
StringReplace("In From Within","In","Out",0,1,0) returns "Out
From Within"  (replaces only the first one)

StringReplace("In From Within","In","Out",0,-1,0) returns
"Out
From without"  (replaces all occurrences)

StringReplace("In From Within","In","Out",1,-1,0) returns
"Out
From Within"  (replaces all that match case)

StringReplace("In From Within","In","Out",0,-1,1) returns
"Out
From Within"  (replaces all that are whole words)
```

Примечание. Функция **StringReplace()** не распознает специальные символы, такие как @#$%&*(). Эта функция воспринимает их как разграничители. Например, при выполнении функции **StringReplace(abc#,abc#,1234,0,1,1)** не произойдет никаких замен. Знак # распознается как разграничитель, а не символ.

StringRight() строка

Возвращает последний (крайний правый) символ(ы) в указанном текстовом тэге.

Синтаксис

```
MessageResult=StringRight("Text",Chars);
```

Параметр	Описание
Text	Фактическая строка или текстовый тэг.
Chars	Количество возвращаемых символов.

Комментарий Если значение *Chars* установлено в 0, будет возвращена вся строка.

Примеры

```
StringRight("The Pump is On", 2) возвращает "On"
StringRight("The Pump is On", 5) возвращает "is On"
StringRight("The Pump is On", 87) возвращает "The Pump is On"
StringRight("The Pump is On", 0) возвращает "The Pump is On"
```

StringSpace() строка

Формирует строку пробелов внутри текстового тэга или выражения.

Синтаксис

```
MessageResult=StringSpace(NumSpaces);
```

Параметр	Описание
NumSpaces	Количество возвращаемых пробелов — константа или целый тэг.

Комментарий Функция **StringSpace()** возвращает строку пробелов указанной (*NumSpaces*) длины.

Примеры Все пробелы представлены символами "ч":

```
StringSpace(4) возвращает "ччччч"
"Pump" + StringSpace(1) + "Station" возвращает "PumpчStation"
```

StringTest() строка

Проверяет, принадлежит ли первый символ строки заданному типу.

Синтаксис

```
DiscreteResult=StringTest("Text",TestType)
```

Параметр	Описание
Text	Строка, которую обрабатывает функция. Фактическая строка или текстовый тэг.
TestType	Определяет одно из следующего:

Тип	Описание
1	Алфавитно-цифровой символ (‘A’-’Z’, ‘a’-’z’ и ‘0-9’)
2	Цифровой символ (‘0-9’)
3	Алфавитный символ (‘A’-’Z’, ‘a’-’z’)
4	Символы в верхнем регистре (‘A’-’Z’)
5	Символы в нижнем регистре (‘a’-’z’)
6	Знаки пунктуации (0x21-0x2F)
7	ASCII символы (0x00-0x7F)
8	Шестнадцатеричные символы (‘A’-’F’ и ‘a’-’f’ и‘0’-’9’)
9	Печатаемые символы (0x20-0x7E)
10	Управляющие символы (0x00-0x1F и 0x7F)
11	Символы пробела (0x09-0x0D и 0x20)

Комментарий

Функция **StringTest()** возвратит положительное число в *DiscreteResult*, если первый символ в строке *Text* принадлежит типу, указанному в аргументе *TestType*. Как и в других функциях, где обрабатывается только один символ, если строка содержит более одного символа, проверен будет только превый.

Примеры

```
StringTest("ACB123",1) возвращает 1
StringTest("ABC123",5) возвращает 0
```

StringToIntg() строка

Преобразует числовое значение в текстовом тэге в целое значение, над которым можно производить математические вычисления.

Синтаксис

```
IntegerResult=StringToIntg("Text");
```

Параметр	Описание
Text	Строка, к которой будет применена функция. Фактическая строка или текстовый тэг.

Комментарий Когда выполняется эта функция, система считывает первый символ в строке, содержащей числовое значение. Если первый символ не является числовым (пробелы игнорируются), значение результата равно нулю (0). Если первый символ оказывается числовым, система продолжает последовательно считывать символы до тех пор, пока не встретится нечисловой символ.

Примеры

```
If Text="ABCD", then IntegerTag=0.
If Text="22.2 is the Value", then IntegerTag=22
(поскольку результат возвращается в тэг целого типа).
If Text="The Value is 22", then IntegerTag=0.
```

См. также StringFromIntg(), StringFromReal(), StringToReal(), Text()

StringToReal() строка

Преобразует числовое значение в текстовом тэге в действительное значение (с плавающей точкой), над которым можно производить математические вычисления.

Синтаксис

```
RealResult=StringToReal("Text");
```

Параметр	Описание
Text	Строка, к которой будет применена функция. Фактическая строка или текстовый тэг.

Комментарий Когда выполняется эта функция, система считывает первый символ в строке, содержащей числовое значение. Если первый символ не является числовым (пробелы игнорируются), значение результата равно нулю (0). Если первый символ оказывается числовым, система продолжает последовательно считывать символы до тех пор, пока не встретится нечисловой символ.

Примеры

```
If Text="ABCD", then RealTag=0.
If Text="22.261 is the Value", then RealTag=22.261.
If Text="The Value is 22", then RealTag=0.
```

См. также StringFromReal(), StingFromIntg(), StringToIntg(), Text()

StringTrim() строка

Удаляет ненужные пробелы из текстового тэга.

Синтаксис

```
MessageResult=StringTrim("Text",TrimType);
```

Параметр	Описание
Text	Строка, которую обрабатывает эта функция. Фактическая строка или текстовый тэг.
TrimType	Определяет один из следующих способов:

Тип	Описание
1	Удалять пробелы слева от первого символа, отличного от пробела.
2	Удалять пробелы справа от последнего символа, отличного от пробела.
3	Удалять все пробелы, кроме одиночных пробелов между словами.

Комментарий

В строке *Text* ищутся пробелы (ASCII 0x09-0x0D и 0x20), которые надо удалить. Аргумент *TrimtType* определяет способ применения этой функции.

Пример

В этом примере все пробелы обозначены символом «x».

StringTrim("ччччThisчisчaччtestчччччч", 1) возвращает "Thisчisчaччtestчччччч"

StringTrim("ччччThisчisчaччtestчччччч", 2) возвращает "ччччThisчisчaччtest"

StringTrim("ччччThisчisчaччtestчччччч", 3) возвращает "ThisчisчaЧtest"

Примечание. Функцию **StringReplace()** можно использовать для удаления ВСЕХ пробелов из указанного текстового тэга. Просто замените все символы пробела на «null».

StringUpper() строка

Преобразует все символы в нижнем регистре указанного тэга типа сообщение в верхний регистр.

Синтаксис

```
MessageResult=StringUpper("Text");
```

Параметр	Описание
Text	Строка, которую обрабатывает функция. Фактическая строка или текстовый тэг.

Комментарий

Преобразование не повлияет на символы в верхнем регистре, цифры, знаки и друие специальные символы.

Примеры

StringUpper("abcd") возвращает "ABCD."

StringUpper("22.2 is the value") возвращает "22.2 IS THE VALUE"

Tan() математическая

Возвращает *тангенс* угла, заданного в градусах.

Синтаксис

```
Result=Tan(AngleNumber);
```

Параметр	Описание
AngleNumber	Угол в градусах. Любое число или тэг действительного или целого типа.

Примеры

```
Wave = 10 + 50 * Tan(6 * $seconds);
```

Tan(45) возвращает 1

Tan(0) возвращает 0

См. также **Sin(), Cos()**

Text() строка

Преобразует числовое значение аналогового тэга (целое или действительное) в строку вывода по указанному формату *Format_Text*.

Синтаксис

```
MessageResult=Text(Analog_Tag,"Format_Text");
```

Параметр	Описание
Analog_Tag	Константа, действительный или целый тэг, который будет преобразован.
Format_Text	Формат, используемый для преобразования. Фактическая строка или текстовый тэг.

Примеры

```
MessageTag=Text(66,"#.00");
```

MessageTag — это текстовый тэг, 66 — целый или действительный тэг, а «#0.00» представляет формат вывода числа.

```
If Analog_Tag=66, then MessageTag=66.00.
If Analog_Tag=22.269, then MessageTag=22.27.
If Analog_Tag=9.999, then MessageTag=10.00.
LogMessage("Текущее значение FreeserRoomTemp:" + Text
(FreezerRoomTemp, "#.#"));
```

В следующем примере MessageTag получит значение "One=1 Two=2".

```
MessageTag = "One + " + Text(1,"#") + StringChar(32) + "Two
+" + Text(2,"#");
```

См. также **StringFromIntg(), StringToIntg(), StringFromReal(), StringToReal()**

Trunc() математическая

Обрезает действительное число простым отбрасыванием дробной части справа от десятичной точки.

Синтаксис

```
ResultNumericTag=Trunc(Number);
```

Параметр	Описание
Number	Любое число или тэг действительного или целого типа.

Комментарий Эта функция дает тот же результат, как простое присваивание действительного тэга целому тэгу.

Примеры

```
Trunc(4.3) возвращает 4
Trunc(-4.3) возвращает -4
```

См. также **Round()**

wcAddItem() элементы управления окна

Добавляет заданную строку в список (List box) или поле ввода со списком (Combo box). Если списки не созданы как сортируемые, строка добавляется в конец списка. В противном случае строка вставляется в список, и список сортируется.

Синтаксис

```
[ErrorNumber=]wcAddItem("ControlName","MessageTag");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
MessageTag	Выводимая текстовая строка. Фактическая строка или текстовый тэг.

Информация о кодах ошибок содержится в Приложении А.

Применение Списки типа List box и Combo box.

Примеры Это выражение добавляет содержимое строки сообщения в список (List box), когда открывается окно (по сценарию открытия окна), содержащее список типа List box:

```
wcAddItem("ListBox_1", "Chocolate");
wcAddItem("ListBox_1", "Vanilla");
wcAddItem("ListBox_1", "Strawberry");
```

См. также **wcInsertItem()**

wcClear() элементы управления окна

Удаляет все строки из списка типа List box или Combo box.

Синтаксис

```
[ErrorNumber=]wcClear("ControlName");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

Пример Это выражение удаляет все строки в списке Combo box, когда нажимается кнопка (с которой связан сценарий типа Action Script):

```
wcClear("ListBox_1");
```

wcDeleteItem() элементы управления окна

Удаляет пункт, связанный с индексным тэгом в списках типа List box и Combo box.

Синтаксис

```
[ErrorNumber=]wcDeleteItem("ControlName",ItemIndex);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
ItemIndex	Номер расположения пункта в списке — константа или целый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

Пример Это выражение удаляет третью строку в списке, когда нажимается кнопка (с которой связан сценарий типа Action Script):

```
wcDeleteItem("ListBox_1", 3);
```

wcDeleteSelection()
элементы управления окна

Удаляет из списка пункт, выбранный в текущий момент.

Синтаксис

```
[ErrorNumber =]wcDeleteSelection("ControlName");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.

☞ Информация о кодах ошибок содержится в Приложении А.

Применение

Списки типа List box и Combo box.

Пример

Удаляет из списка строку, выбранную в текущий момент.

```
wcDeleteSelection("ListBox_1");
```

wcErrorMessage()
элемент управления окна

Возвращает строку сообщения, описывающую ошибку.

Синтаксис

```
ErrorMessage=wcErrorMessage(ErrorNumber);
```

Параметр	Описание
ErrorMessage	Текстовый тэг.
ErrorNumber	Числовой код ошибки, возвращаемый всеми элементами управления Windows. Константа или целый тэг.

☞ Информация о кодах ошибок содержится в Приложении А.

Применение

Списки типа List box и Combo box, поля ввода текста (Text box), флажки (Check box) и переключатели (Radio button).

Примеры

Если ошибка возникает во время загрузки списка, выведите текстовое описание ошибки в тэг-сообщение *ErrorDescription*. В данном примере анимационная связь вывода строки *StringValueOutput* была назначена тэгу *ErrorDescription* для вывода на дисплей сообщения об ошибке.

Сценарий типа On Show Window (при открытии окна):

```
ErrorNumber=wcLoadList("ListBox_1","c:\InTouch\recipe.txt");
ErrorDescription=wcErrorMessage(errornumber);
```

Эта функция может служить для вывода сообщений об ошибках при выполнении всех функций элементов управления Windows:

```
ErrorNumber=wcAddItem("ListBox_1","United States");
ErrorMsg=wcErrorMessage(ErrorNumber);
```

wcFindItem() элементы управления окна

Возвращает соответствующий индекс первого пункта в списках типа List box и Combo box, который совпадает со строкой, указанной в аргументе *Message*.

Синтаксис

```
[ErrorNumber=]wcFindItem
("ControlName","MessageTag",DiscreteTag,
Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
MessageTag	Сравниваемая строка. Фактическая строка или текстовый тэг.
DiscreteTag	Определяет тип сравнения строки. Можно присвоить одно из следующих значений: 0=без учета регистра 1=с учетом регистра
Tagname	Фактическое имя целого типа.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

wcGetItem()элементы управления Windows

Возвращает значение строки пункта, связанного с соответствующим индексом ItemIndex в списках типа List box и Combo.

Синтаксис

```
[ErrorNumber=]wcGetItem("ControlName",ItemIndex,Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
ItemIndex	Номер расположения пункта в списке — константа или целый тэг.
Tagname	Фактическое имя действительного или целого тэга. Функция **wcGetItem** поместит в этот тэг числовое значение соответствующего пункта.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

Пример Это выражение возвращает значение строки 10-го пункта списка Combo box в текстовый тэг *ListSelection*, когда нажимается кнопка (сценарий действия по нажатию кнопки):

```
wcGetItem("Combobox_1", 10, ListSelection);
```

Если 10-ый пункт — «Vanilla», то тэг *ListSelection* будет содержать строку «Vanilla».

wcGetItemData() элементы управления Windows

Возвращает целое значение, связанное с соответствующим индексом *ItemIndex* в списках типа List box и Combo box.

Синтаксис

```
[ErrorNumber=]wcGetItemData("ControlName",ItemIndex,Tagname);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
ItemIndex	Число, соответствующее расположению пункта в списке — константа или целый тэг.
Tagname	Фактическое имя действительного или целого тэга. Функция **wcGetItemData** поместит в этот тэг числовое значение соответствующего пункта.

☞ Информация о кодах ошибок содержится в Приложении А.

Применение

Списки типа List box и Combo box.

Пример

Это выражение возвращает числовое значение, связанное с 5-ым пунктом списка List box, в тэг целого типа *ItemValue*, когда нажимается кнопка (сценарий действия по нажатию кнопки):

```
wcGetItemData("ListBox_1", 5, ItemValue);
```

Если 5-му пункту списка присвоено значение 4500, то тэг *ItemValue* будет содержать 4500.

См. также

wcSetItemData()

wcInsertItem() элементы управления Windows

Вставляет строку *Message* в список. Индекс *ItemIndex* соответствует позиции в списке List box, в которую будет помещена строка. В отличие от функции **wcAddItem()** функция **wcInsertItem()** не будет сортировать список, даже если он был создан как сортируемый.

Синтаксис

```
[ErrorNumber=]wcInsertItem("ControlName",ItemIndex,"MessageTag");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
ItemIndex	Число, соответствующее расположению вставляемого пункта. Если этот параметр имеет значение -1, строка добавляется в конец списка — константа или целый тэг.
MessageTag	Содержимое вставляемой строки. Фактическая строка или текстовый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

Пример Это выражение вставляет новый пункт под названием «Blueberry»в список List box на 4-ую позицию сверху, когда нажимается кнопка (сценарий действия по нажатию кнопки):

```
wcInsertItem("ListBox_1", 4, "Blueberry");
```

См. также **wcAddItem()**

wcLoadList() элементы управления Windows

Заменяет содержимое списков List box или Combo box строками, содержащимися в указанном файле.

Синтаксис

```
[ErrorNumber=]wcLoadList("ControlName","Filename");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
Filename	Файл, содержащий данные в чистом ASCII формате. Если в этом параметре не указан полный путь файла, то функция ищет его в каталоге приложения. Фактическая строка или текстовый тэг. ☞ Информация о кодах ошибок содержится в Приложении A.

Применение Списки типа List box и Combo box.

Пример Это выражение загружает правильно отформатированную информацию (расположенную в файле c:\InTouch.32\wclist.txt) в список Combo box, когда

открывается окно (в сценарии типа On Show Window Script), содержащее Combo box:

```
wcLoadList("Combobox_1", "c:\InTouch.32\wclist.txt");
```

См. также **wcAddItem(), wcSaveList()**

wcLoadText()элементы управления Windows

Заменяет содержимое объекта Text box строками, содержащимися в указанном файле.

Синтаксис

```
[ErrorNumber=]wcLoadText("ControlName", "Filename");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
Filename	Файл, содержащий данные в чистом ASCII формате. Если в этом параметре не указан полный путь файла, то функция ищет его в каталоге приложения. Фактическая строка или текстовый тэг.

Комментарий

Функция **wcLoadText()** поддерживает только чисто ASCII файлы, создаваемые, например, программой Microsoft Notepad. Коды ошибок для элементов управления Windows приведены в Приложении A.

📖 Расширенные возможности просмотра файлов предусмотрены в программе просмотра из комплекта Factory Suite *Productivity Pack*.

Применение

Поля ввода текста (Text box).

Пример

Это выражение загружает текстовый файл, созданный программой Notepad, (c:\InTouch.32\readme.txt) в объект Text box, когда открывается окно, содержащее Text box (в сценарии типа On Show Window Script):

```
wcLoadText("Textbox_1", "c:\InTouch.32\readme.txt");
```

wcSaveList() элементы управления Windows

Заменяет содержимое указанного файла записями в списке List box или Combo box

Синтаксис

```
[ErrorNumber=]wcSaveList("ControlName","Filename");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
Filename	Файл, содержащий данные в чистом ASCII формате. Если файл не существует, он создается. Записи из файла могут быть впоследствии загружены в список с помощью функции **wcLoadList()**. Фактическая строка или текстовый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение

Списки типа List box и Combo box.

Пример

Это выражение сохраняет текущие записи из списка Combo box в файл записи (c:\InTouch.32\newlist.txt), когда нажимается кнопка (сценарий действия по нажатию кнопки):

```
wcSaveList("ListBox_1", "c:\InTouch.32\newlist.txt");
```

Примечание. Если для заполнения списков List box и Combo box используются внешние ASCII файлы, они должны быть подготовлены в специальном формате и содержать указанную инфрмацию. Формат:

ControlType, ListCount
ListItem, ItemData
ListItem, ItemData
: :
: :
ListItem, ItemData

Пример: Файл должен быть загружен в список Combo box. Он содержит три строки (пункта) для выбора, которым пока не присвоено никаких данных. (См. описание функции **wcSetItemData()** о формате информации в списках.) Формат файла будет выглядеть следующим образом:

COMBOBOX, 3
Chocolate, 0
Vanilla, 0
Strawberry, 0

Описание: COMBOBOX — тип элемента управления. ListCount равен 3, поскольку содержит 3 пункта в списке: Chocolate, Vanilla и Strawberry. Chocolate является первым ListItem или имеет индекс 1. Vanilla имеет индекс 2, а Strawberry — 3. У каждого из этих пунктов значение данных равно 0.

См. также

wcLoadList(), wcSetItemData()

wcSaveText()элементы управления Windows

Сохраняет текст, содержащийся в Text box, в файл, имя которого указано аргументом *FileName*. Если файл не существует, он будет создан. Если он создан, он должен иметь атрибут «на чтение и запись».

Синтаксис

```
[ErrorNumber=]wcSaveText("ControlName","Filename");
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
Filename	Файл, содержащий данные в чистом ASCII формате. Если не указан полный путь, функция будет искать его в директории приложения. Если файл не существует, он создается. Пункты могут затем загружены в список с помощью функции **wcLoadList()**. Фактическая строка или текстовый тэг.

☞ Информация о кодах ошибок содержится в Приложении A.

Применение Поле ввода текста (Text box).

Пример Это выражение сохраняет текущую информацию из объекта Text box в файл c:\InTouch.32\newtext.txt, когда нажимается кнопка (сценарий действия по нажатию кнопки):

```
wcSaveText("Textbox_1", "c:\InTouch.32\newtext.txt");
```

Примечание. Функция сохраняет данные только в ASCII файлы, созданные, например, с помощью программы Notepad.

См. также **wcLoadList()**

wcSetItemData() элементы управления Windows

Присваивает целое значение (*Number*) пункту списка, заданного аргументом *ItemIndex*. Эта функция позволяет присваивать целое число строке.

Синтаксис

```
[ErrorNumber=]wcSetItemData("ControlName",ItemIndex,Number);
```

Параметр	Описание
ControlName	Имя элемента управления Windows, например, *ListBox_1*. Фактическая строка или текстовый тэг.
ItemIndex	Целое значение, задающее порядковый номер требуемого пунтктa — константа или целый тэг.
Number	Целое значение, присваиваемое пункту — константа или целый тэг.

Комментарий

Списки пунктов могут быть созданы во внешней программе (напр., в Блокноте), а затем загружены за один вызов функции. Список должен быть отформатирован, как показано в описании функции **wcSaveList()**.

☞ Информация о кодах ошибок содержится в Приложении A.

Примеры

Дан рецепт с тремя ингредиентами: мука, сахар и соль. Количество муки 4500 грамм, сахара 1500 и соли 325 грамм. Значения присваиваются каждому пункту списка с помощью сценария типа Data Change Script, который выполняется при изменении тэга ReciprName, определяющего, какой рецепт выбран в данный момент:

```
wcSetItemData("ListBox_1", 1, 4500); {устанавливает значение
первого пункта (мука)=4500}
wcSetItemData("ListBox_1", 2, 1500); {устанавливает значение
второго пункта (сахар)=1500}
wcSetItemData("ListBox_1", 3, 325);   {устанавливает значение
третьего пункта (соль)=325}
```

Функция **wcGetItemData()** используется для возврата значения пункта (данных пункта) в аргумент *Tagname*. Этот аргумент может быть целым тэгом DDE, производящим запись напрямую в действующее устройство.

См. также **wcLoadList(), wcSaveList(), wcGetItemData()**

WWControl() прочие

Позволяет управлять из InTouch другим приложением — восстанавливать, сворачивать, разворачивать и закрывать окно приложения.

Синтаксис

```
WWControl("AppTitle","ControlType");
```

Параметр	Описание
AppTitle	Название заголовка приложения. Заголовок приложения может быть определен с помощью функции **InfoAppTitle()**. Фактическая строка или текстовый тэг.
ControlType	Определяет один из следующих способов управления приложением (Эти действия идентичны нажатию на соответствующий пункт управляющего меню приложением). Фактическая строка или текстовый тэг.

Тип	Описание
"Restore"	Делает активным и выводит на дисплей окно приложения.
"Minimize"	Делает активным и выводит окно на дисплей в виде пиктограммы (значка).
"Maximize"	Делает активным и выводит окно приложения на весь экран.
"Close"	Закрывает приложение.

Примеры

```
WWControl("Calculator","Restore");
WWControl("Microsoft Excel","Close");
WWControl(InfoAppTitle("View"), "Close");
```

См. также **InFoAppTitle(), ActivateApp(), StartApp()**

WWExecute() WWDDE

Посылает команду (используя команду DDE Execute) указанному приложению *Application* и теме *Topic*.

Синтаксис

```
[Status=]WWExecute("Application","Topic","Command");
```

Параметр	Описание
Application	Приложение, которому посылается команда. Фактическая строка или текстовый тэг.
Topic	Тема внутри приложения, которой посылается команда. Фактическая строка или текстовый тэг.
Command	Посылаемая команда. Фактическая строка или текстовый тэг.

Комментарий

Командная строка *Command* будет послана конкретно указанному приложению *Application* и теме *Topic*.

Примеры

Это выражение выполняет макрокоманду в Excel:

```
Macro="Macro1!TestMacro";
Command="[Run(" + StringChar(34) + Macro + StringChar(34) +
",0)]";
WWExecute("excel","system",Command);
```

В результате выполнения функции WWExecute(«excel», «system», Command) программе Excel будет послано следующее (и будет запущен *TestMacro*):

```
[Run("Macro1!TestMacro")]
```

В следующем примере выполняется макрос в Microsoft Access:

```
WWExecute("MSAccess","system","MyMacro");
```

Функция **WWExecute()** возвращает 1, если приложение запущено, тема существует и команда была послана успешно. Она возвращает 0, если приложение занято и **-1** при возникновении ошибки. Таким образом, можно отследить состояние выполнения команды:

```
Status=WWExecute("excel","system",Command);
```

Status это целый тэг, в который записываются значения 1, -1 или 0.

WWPoke() WWDDE

Посылает значение (используя команду DDE Poke) указанному приложению *Application*, теме *Topic* и элементу *Item*.

Синтаксис

```
[Status=]WWPoke("Application","Topic","Item","TextValue");
```

Параметр	Описание
Application	Приложение, которому посылается команда Poke. Фактическая строка или текстовый тэг.
Topic	Тема внутри приложения, которой посылается команда Poke. Фактическая строка или текстовый тэг.
Item	Элемент внутри темы, куда записывается значение. Фактическая строка или текстовый тэг.
TextValue	Текстовая переменная или строка. Если значение, которое вы хотите послать, является числом, его можно преобразовать с помощью функций **Text()**, **StringFromIntg()** или **StringFromReal()**. Фактическая строка или текстовый тэг.

Комментарий

Значение *TextValue* будет послано конкретно указанному приложению *Application*, теме *Topic* и элементу *Item*.

Пример

Это выражение преобразует значение в текст и записывает его в ячейку таблицы Excel:

```
String=Text(Value,"0");
WWPoke("excel","sheet1","r1c1",String);
```

📖 Подробная информация об использовании Excel 5.0 с InTouch (DDE) содержится в *"Руководстве пользователя InTouch"*.

Примечание. Действие функции **WWPoke()** из одного приложения "View" в другое "View" не определено и не поддерживается. Успешное выполнение WWPoke() в этом случае не гарантируется.

Функция **WWPoke()** возвращает 1, если приложение запущено, тема и элемент существуют и значение было послано успешно. Она возвращает 0, если приложение занято и -1 при возникновении ошибки. Таким образом, можно отследить состояние выполнения команды:

```
Status=WWPoke("excel","sheet1","r1c1",String);
```

Status — это целый тэг, в который записываются значения 1, -1 или 0.

См. также **Text(), StringFromIntg(), StringFromReal()**

WWRequest() WWDDE

Делает однократный запрос значения (используя команду DDE Request) из указанного приложения *Application*, темы *Topic* и элемента *Item*.

Синтаксис

```
WWRequest(Application,Topic,Item,ValueMsg_Tag);
```

Параметр	Описание
Application	Приложение, у которого запрашиваются данные. Фактическая строка или текстовый тэг.
Topic	Тема внутри приложения, у которой запрашиваются данные. Фактическая строка или текстовый тэг.
Item	Элемент внутри темы, откуда запрашиваются данные. Фактическая строка или текстовый тэг.
ValueMsg_Tag	Тэг текстового типа, куда возвращаются запрошенные данные. Фактическая строка или текстовый тэг.

Комментарий

Значение DDE из конкретно указанного приложения *Application*, темы *Topic* и элемента *Item* будет записано в тэг *ValueMsg_Tag*.

Значение будет возвращено как строка. Если это число, его можно преобразовать с помощью функций **StringToIntg()** и **StringToReal()**.

Примеры

Это выражение запрашивает значение из ячейки таблицы Excel и преобразует полученную строку в число:

```
WWRequest("excel","[Book1.xls]sheet1","r1c1",Result);
Value=StringToReal(Result);
```

Функция **WWRequest()** возвращает 1, если приложение запущено, тема и элемент существуют и значение было получено успешно. Она возвращает 0, если приложение занято и -1 при возникновении ошибки. Таким образом, можно отследить состояние выполнения команды:

```
Status=WWRequest("excel","[Book1.xls]sheet1","r1c1",Result);
```

Status — это целый тэг, в который записываются значения 1, -1 или 0.

См. также **StringToIntg(), StringToReal()**

ПРИЛОЖЕНИЕ А

Отладка функций Quick-сценариев

Некоторые функции сценариев InTouch возвращают значения, зависящие от результата выполнения функции. По этому значению (коду результата) можно определять, насколько было успешным выполнение функции. В этом приложении описываются коды результата для функций элементов управления Windows, рецептурных и SQL функций. Дано также описание DDE-элементов системы SPC (статистического управления процессом), которые можно использовать для получения информации о наборах данных (Dataset) и для управления выводом графов на экран.

Сообщения об ошибках элементов управления окна и распределенных алармов

Функции элементов управления окна и распределенных алармов возвращают значение, основанное на результате обработки функции сценария. Возвращаемое значение можно присвоить целому тэгу для диагностики ошибок. Например:

```
ErrorNumber = wcGetItem("ControlName", Number, Tagname);
```

Где:

ErrorNumber — это тэг целого типа, куда будет помещено возвращаемое значение ошибки. Значение, возвращаемое функцией, можно передать в качестве аргумента функции **wcErrorMessage()**. Функция **wcErrorMessage()** возвратит строку описания этой ошибки. Например:

```
ErrorMsg = wcErrorMessge(ErrorNumber);
```

Где:

ErrorMsg — это тэг текстового типа, который содержит текст описания возвращенной ошибки. В приведенной ниже таблице указаны числовые значения ошибок и их описание:

Сообщение	Описание
0	Успешно
-1	Общая ошибка
-2	Недостаточно памяти
-3	Свойство с атрибутом «только чтение»
-4	Указанный элемент данных уже существует
-5	Неизвестное имя объекта
-6	Неизвестное имя свойства
-*x**	Неизвестная ошибка

* -*x* обозначает любое другое число.

Чтобы получить код ошибки от рецептурных функций, он должен быть приравнен какому-либо целому тэгу InTouch.

Пример:

```
ErrorCode = RecipeLoad(FileName, UnitName, RecipeName);
```

Функция **RecipeLoad()** установит значение тэга **ErrorCode** в **0**, если она выполнилась успешно. Если не удалось выполнить **RecipeLoad()**, она присвоит значению аналогового тэга **ErrorCode** число, соответствующее конкретному условию ошибки. Ниже перечислены возможные коды ошибок и их описание:

Значение	Сообщение об ошибке	Описание
0	**Успешно**	Вызванная рецептурная функция выполнена успешно.
-1	**Нет такого шаблона рецепта**	Указанный файл рецептурных шаблонов не существует.
-2	**Не активен режим просмотра**	Рецептурная функция, пущенная другой программой, не может быть выполнена, потому что не запущен **WindowViewer**.
-3	**Недостаточно памяти**	Недостаточно памяти для выполнения текущего задания.
-4	**Слишком длинная строка в файле шаблона рецепта**	Строка в файле рецептурных шаблонов превысила максимально допустимую длину.
-5	**Усеченная строка в шаблоне рецепта**	Строка в файле рецептурных шаблонов обрезана.
-6	**Неверный файл шаблона рецепта**	Указанный файл не является допустимым файлом рецептурных шаблонов в формате CSV. (См. главу 2 «Настройка шаблонов рецептов» в *"Руководстве пользователя по Recipe Manager"*.)
-7	**Требуется "unit" или "recipe"**	Указанное имя блока или рецепта не найдено в файле шаблона рецепта. См. *"Руководстве пользователя по Recipe Manager"*.
-9	**Имя рецепта не найдено в файле шаблона рецепта**	Указанное имя рецепта не определено в файле рецептурных шаблонов.
-10	**Имя блока не найдено в файле шаблона рецепта**	Указанное имя блока не определено при описании блоков в файле рецептурных шаблонов.
-12	**Требуется \»Analog»,\»Discrete»,**	Некорректный тип введен для элемента данных в файле рецептурных шаблонов.

Значение	Сообщение об ошибке »Message»	Описание
		Допустимыми являются только типы **Analog**, **Discrete** и **Message**.
-13	**Не совпадает тип тэга \»Analog»,\»Discrete»,\ »Message»**	Указан некорректный тэг для данного типа элемента данных, например, элемент данных рецепта определен как **Analog**, а в блоке для соединения с ним указан тэг типа **Message**.
-14	**Неверное дискретное значение, требуется \»0»,\»1»**	Некорректное значение было введено для **дискретного** тэга в файле рецептурных шаблонов. Для **дискретных** тэгов допустимы только значения **0** и **1**.
-15	**Не удается открыть временный файл**	Не удается открыть временный файл; возможно, из-за недостаточного дискового пространства.
-16	**Ошибка записи при сохранении файла шаблона рецепта**	При записи файла рецептурных шаблонов возникла ошибка.
-17	**Пользователь не сделал выбор**	Пользователь выбрал кнопку **Отмена** в диалоговом окне **выбора рецептов**, вместо того, чтобы выбрать имя рецепта.
-19	**Шаблон рецепта занят другим приложением**	Указанный файл рецептурных шаблонов открыт, и следовательно, **WindowViewer** не имеет к нему доступа.

Вывод сообщений с кодами ошибок

Как было описано выше, каждая рецептурная функция возвращает число, характеризующее ошибочное состояние для функции. Используя функцию **RecipeGetMessage()** в сценарии изменения данных InTouch (Data Change script), соответствующий код ошибки можно записать в аналоговый тэг, а связанное с кодом сообщение об ошибке можно записать в текстовый тэг.

Для этого можно использовать следующее выражение сценария:

```
RecipeGetMessage(ErrorCode, ErrorMessage, 131);
```

Это выражение сценария будет автоматически выполняться всякий раз, когда будет меняться значение аналогового тэга **ErrorCode**. При выполнении этого выражения функция **RecipeGetMessage()** считает текущее числовое значение тэга **ErrorCode** и возвратит связанное с этим значением сообщение в тэг **ErrorMessage**.

Имена элементов SPC DDE

Для получения информации о наборах данных и для управления операциями вывода диаграмм и графов имеются специальные DDE-элементы. Именем приложения (*Application*) является SPC. Именем темы (*Topic*) будет имя набора данных (*Dataset*).

Примечание. Все DDE-элементы SPC используются одинаково для распределенного и нераспределенного SPC, за исключением того, что имя темы (*Topic*) для распределенного SPC должно быть косвенным именем набора данных (*Indirect Dataset Name*).

Элементы управления и вывода на экран SPC DDE

DDE-элементы управления и вывода на экран используются для управления и вывода на дисплей информации о наборе данных (Dataset). Можно осуществлять изменения накопленных и выводимых на дисплей данных о продукте для локальных и удаленных наборов данных. Изменения, производимые путем нажатия кнопкой мышки на объекте дисплея, влияют на выводимый на дисплей продукт. А DDE-элементы SPC изменяют накапливаемый продукт. Выводимый на дисплей продукт и накапливаемый продукт имеют свои собственные текущие выборки, являющиеся наиболее последними записанными выборками.

DDE-элементы *управления* доступны для всех узлов. Это значения для набора данных накапливаемого продукта для удаленной базы данных. Имя набора данных всегда включает в себя имя узла. DDE-элементы *вывода на экран* являются локальными для каждого узла. Это значения выборки для выводимого на дисплей продукта для каждого узла.

Алармы вычисляются и хранятся для накапливаемого и выводимого продукта. О них сообщается только в режиме исполнения для накапливаемого продукта.

Примечание. При добавлении продуктов для вывода и накопления, многие DDE-элементы остаются применимы только к текущему накапливаемому продукту. Эти элементы отмечены в приведенном ниже списке знаком звездочки (*) перед именем DDE-элемента SPC.

Имя элемента	Тип	Доступ	Описание
AutoCollection	Дискретный	ЧЗ	Разрешает и запрещает автоматическое накопление данных.
***CalculateControlLimits**	Дискретный	ЧЗ	Устанавливает в 1 значение предела начала вычислений.
DatasetName	Текстовый (32)	ЧЗ	Выводит на дисплей имя файла конфигурации групп данных.
HistogramLCL	Действительный	ТЧ	Выводит нижний предел управления гистограммы на основе генеральной совокупности.
HistogramUCL	Действительный	ТЧ	Выводит верхний предел управления гистограммы на основе генеральной совокупности.
Kurtosis	Действительный	ТЧ	Вид распределения гистограммы.
LastSampleDisplayed	Целый	ЧЗ	Устанавливает номер последней выборки, выводимой набором данных Dataset.
***ManualInputDialog**	Дискретный	ЧЗ	Устанавливает значение 1 для вывода на дисплей встроенного диалогового окна ручного ввода.
MeasurementsPerSample	Целый	ТЧ	Выводит сконфигурированное число измерений на одну выборку.
NewProduct	Текстовый (32)	ЧЗ	Используется для создания нового имени продукта.
***ProductCollected**	Текстовый (32)	ЧЗ	Изменяет имя продукта, накапливаемого в наборе данных Dataset.
ProductDisplayed	Текстовый (32)	ЧЗ	Изменяет имя продукта, выводимого набором данных Dataset.
SampleSize	Целый	ТЧ	Размер выборки для NP Dataset.
SamplesPerControlChart	Целый	ЧЗ	Устанавливает количество выборок, выводимых в контрольном графе.
SamplesPerHistogram	Целый	ЧЗ	Устанавливает количество выборок, выводимых в гистограмме.
SamplesPerLimitCalc	Целый	ЧЗ	Устанавливает количество выборок, используемых в вычислении пределов регулирования.
SamplesPerPareto	Целый	ЧЗ	Устанавливает количество

Имя элемента	Тип	Доступ	Описание
			выборок, используемых при выводе диаграммы Парето.
SelSPCOutSpecMsg	Текстовый	ТЧ	Тэг сообщения аларма "Выборка вне предела спецификации".
Skewness	Действительный	ТЧ	Выводит отклонение от среднего на гистограмме.
SPCAllowSampDelMod	Дискретный	ЧЗ	Включает или выключает команды Удалить и Изменить, доступные при нажатии правой кнопкой на выборку.
StartCollection	Дискретный	ЧЗ	Устанавливается в 1 для запуска цикла автоматического накопления данных.

Элементы текущей выборки SPC DDE

Все DDE-элементы текущей выборки относятся к последней накопленной выборке заданного набора данных. Их можно использовать для изменения необработанных данных и пределов, связанных с именем этого набора данных. Для изменения информации по текущей выборке необходимо записать значение в соответствующий DDE-элемент и установить значение 1 DDE-элементу **CurrentUpdate**. Результат будет таким же, как при повторном вводе выборки, и это вызовет выполнение всех требуемых вычислений. Программа SPC сбросит значение **CurrentUpdate** в 0 после ввода выборки. Но если начался цикл накопления следующей выборки, элементы текущей выборки не могут быть обновлены.

DDE-элементы текущей выборки доступны для всех узлов. Значения текущей выборки представляют собой последнюю накопленную выборку накапливаемого продукта.

Для распределенных систем SPC первоначально все значения устанавливаются в ноль. SPC соединяется с удаленными наборами данных и опрашивает все данные. Значения элементов данных обновляются всегда, когда от удаленного узла приходит новая информация. Измененные значения текущей выборки хранятся в буфере локального узла, пока не будет установлено значение 1 в **CurrentUpdate**. Тогда значения помещаются в пакет текущей выборки и посылаются на удаленный узел для анализа и хранения. В случае, если изменения текущей выборки относятся к другому продукту или если номер текущей выборки не является номером последней записанной выборки, то сервер откажется выполнять этот запрос на обновление и посылку данных.

Примечание. При добавлении продуктов для вывода и накопления, элементы «текущей» выборки остаются применимы только к текущему накапливаемому продукту.

Имя элемента	Тип DDE	Доступ	Описание
CurrentCauseCode	Целый	ЧЗ	Устанавливает код специальной причины для текущей выборки.
CurrentCauseString	Текстовый (128)	ТЧ	Выводит описание кода специальной причины для текущей выборки.
CurrentComment	Текстовый (50)	ЧЗ	Используется для чтения и записи любых смешанных комментариев для текущей выборки.
CurrentCp	Действительный	ТЧ	Выводит выборочный момент текущей выборки.
CurrentCpk	Действительный	ТЧ	Выводит центральный выборочный момент текущей выборки.
CurrentDate	Текстовый (8)	ЧЗ	Устанавливает дату для текущей выборки в формате ГГ-ММ-ДД. Если дата введена некорректно, принимается текущая дата.
CurrentFlag	Дискретный	ЧЗ	Устанавливает флаг для текущей выборки.

Имя элемента	Тип DDE	Доступ	Описание
CurrentIgnoreValue	Дискретный	ЧЗ	Устанавливает режим, при котором текущая выборка будет игнорироваться, если вывод контрольного графа находится в режиме AutoScaled.
CurrentM*x*	Действительный	ЧЗ	Устанавливает номер индивидуального измерения для текущей выборки (x= от 1 до 25)
CurrentR	Действительный	ТЧ	Выводит диапазон текущей выборки
CurrentRBar	Действительный	ЧЗ	Устанавливает средний диапазон текущей выборки.
CurrentRLCL	Действительный	ЧЗ	Устанавливает диапазон нижних пределов управления.
CurrentRUCL	Действительный	ЧЗ	Устанавливает диапазон верхних пределов управления.
CurrentSample	Действительный	ТЧ	Выводит значение последней точки выборки (например, X, C, P).
CurrentSampleBar	Действительный	ЧЗ	Устанавливает среднее текущей выборки в этой точке выборки.
CurrentSampleNumber	Целый	ТЧ	Выводит последнее число измерений в накопленной выборке.
CurrentTarget	Действительный	ЧЗ	Устанавливает значение задания в этой точке выборки.
CurrentTime	Текстовый (8)	ЧЗ	Устанавливает время для текущей выборки в формате ЧЧ-ММ-СС. Если время введено некорректно, принимается текущее время.
CurrentUpdate	Дискретный	ЧЗ	Для изменения информации, вводимой о выборке в любое текущее поле, установите этот элемент в 1.
CurrentXLCL	Действительный	ЧЗ	Устанавливает нижний предел управления для текущей выборки.
CurrentXLSL	Действительный	ЧЗ	Устанавливает нижний предел характеристики текущей выборки.
CurrentXUCL	Действительный	ЧЗ	Устанавливает верхний предел управления для текущей выборки.
CurrentXUSL	Действительный	ЧЗ	Устанавливает верхний предел характеристики текущей выборки.
SPC2L3Out2SD	Целый	ТЧ	Счетчик аларма "2 из последних 3 выборок вне 2 стандартных отклонений SS".
SPC2L3Out2SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "2 из последних 3 выборок вне 2 стандартных отклонений SS".
SPC4L5Out1SD	Целый	ТЧ	Счетчик аларма "4 из последних 5

Имя элемента	Тип DDE	Доступ	Описание
			выборок вне 1 стандартного отклонения SS".
SPC4L5Out1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "4 из последних 5 выборок вне 1 стандартного отклонения SS".
SPCConSampAltUpDn	Целый	ТЧ	Счетчик аларма "Последующие выборки варьируются вверх и вниз".
SPCConSampAltUpDnMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки варьируются вверх и вниз".
SPCConSampln1SD	Целый	ТЧ	Счетчик аларма "Последующие выборки внутри 1 стандартного отклонения".
SPCConSampln1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки внутри 1 стандартного отклонения".
SPCConSamplncDec	Целый	ТЧ	Счетчик аларма "Последующие выборки уменьшаются или увеличиваются".
SPCConSamplncDecMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки уменьшаются или увеличиваются".
SPCConSampOneSideCL	Целый	ТЧ	Счетчик аларма "Последующие выборки по одну сторону от центральной линии".
SPCConSampOneSideCLMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки по одну сторону от центральной линии".
SPCConSampOut1SD	Целый	ТЧ	Счетчик аларма "Последующие выборки вне 1 стандартного отклонения".
SPCConSampOut1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки вне 1 стандартного отклонения".
SPCNLNOutNSD	Целый	ТЧ	Счетчик аларма "? из последних ? выборок вне ? стандартных отклонений".
SPCNLNOutNSDMsg	Текстовый	ТЧ	Тэг сообщения аларма "? из последних ? выборок вне ? стандартных отклонений".
SPCNLNOutNSDSS	Целый	ТЧ	Счетчик аларма "? из последних ? выборок вне ? стандартного отклонения SS".
SPCNLNOutNSDSSMsg	Текстовый	ТЧ	Тэг сообщения аларма "? из последних ? выборок вне ? стандартного отклонения SS".
SPCOutRCtrl	Целый	ТЧ	Счетчик аларма графа диапазона "Диапазон вне контрольного предела".

Имя элемента	Тип DDE	Доступ	Описание
SPCOutRCtrlMsg	Текстовый	ТЧ	Тэг сообщения аларма графа диапазона "Диапазон вне контрольного предела".
SPCOutXCtrl	Целый	ТЧ	Счетчик аларма графа X "Выборка вне контрольного предела".
SPCOutXCtrlMsg	Текстовый	ТЧ	Тэг сообщения аларма графа X "Выборка вне контрольного предела".
SPCOutSpec	Целый	ТЧ	Счетчик аларма "Выборка вне предела спецификации".
SPCOutSpecMsg	Текстовый	ТЧ	Тэг сообщения аларма "Выборка вне предела спецификации".
SPCResetAlarmCounters	Дискретный	ЧЗ	Сброс всех счетчиков алармов.

Элементы ручного ввода SPC DDE

DDE-элементы ручного ввода применяются для создания пользовательских окон ручного ввода. Чтобы можно было воспользоваться DDE-элементами ручного ввода, установите значения соответствующих элементов, а затем установите значение 1 элементу **MI_Save**. При этом информация из других полей ручного ввода (MI - Manual Input) будет введена как новая выборка. Программа SPC сбросит в 0 значение **MI_Save** после ввода выборки..

При распределенном управлении SPC элементы ручного ввода являются частными для каждого узла. Значения сохраняются в локальном буфере на каждом узле до тех пор, пока значение **MI_Save** не установится в 1. В этот момент значения помещаются в пакет ручного ввода и отсылаются на удаленный узел набора данных для анализа и хранения.

Примечание. При добавлении продуктов для вывода и накопления, все элементы ручного ввода остаются применимы только к текущему накапливаемому продукту.

Имя элемента	Тип DDE	Доступ	Описание
MI_CauseCode	Целый	ЧЗ	Устанавливает код специальной причины для ручного ввода измерения.
MI_CauseString	Текстовый (127)	ТЧ	Выводит описание кода специальной причины для ручного ввода измерения.
MI_Comment	Текстовый (50)	ЧЗ	Используется для чтения и записи любых смешанных комментариев для ручного ввода измерения.
MI_Date	Текстовый (8)	ЧЗ	Устанавливает дату ручного ввода измерения в формате ГГ-ММ-ДД. Если дата введена некорректно, принимается текущая дата.
MI_Flag	Дискретный	ЧЗ	Устанавливает флаг для ручного ввода измерения.
MI_IgnoreValue	Дискретный	ЧЗ	Устанавливает режим, при котором ручной ввод измерения будет игнорироваться, если контрольный граф выводится в режиме AutoScaled.
MI_M*x*	Действитель-ный	ЧЗ	Устанавливает номер для ручного ввода измерения (x= от 1 до 25).
MI_Save	Дискретный	ЧЗ	Сохраняет информацию, введенную вручную, в любое поле MI как новую выборку.

Примечание. Когда элемент **MI_Save** установлен в 1, значения всех полей ручного ввода записываются в

Имя элемента	Тип DDE	Доступ	Описание
			соответствующие элементы **Current** и значение **CurrentSampleNumber** возрастает на 1.
MI_Time	Текстовый (8)	ЧЗ	Устанавливает время для ручного ввода измерения в формате ЧЧ-ММ-СС. Если время введено некоррект-но, принимается текущее время.

Элементы выбора SPC DDE

Все DDE-элементы выбора можно использовать для просмотра подробной информации о любой выборке. Элемент **Selection** используется для ввода номера выборки, которую надо вывести на дисплей. Когда этот этот номер будет введен, программа SPC обновит все другие поля подробной информацией для выборки с номером, указанным в элементе **Selection**. Старые данные нельзя изменить, но можно дополнить элементы **SelectionCauseCode** (Код специальной причины), **SelectionFlag** (Флаг) и **SelectionComment** (Комментарий), установив в них соответствующие значения, а затем установив 1 в **SelectionUpdate**. Это вызовет обновление записи выбранной выборки новыми значениями. Программа SPC сбросит значение **SelectionUpdate** в 0 после ввода обновленной выборки.

При распределенном управлении SPC элементы выбора являются частными для каждого узла. Это значения выборки, записанные на удаленном узле для указанного номера выборки накапливаемого продукта. Когда элементу Selection задается номер выборки, информация о выборке считывается из файла с выборкой на удаленном узле. Старые данные нельзя изменить, но можно дополнить элементы **SelectionCauseCode** (Код специальной причины), **SelectionFlag** (Флаг) и **SelectionComment** (Комментарий), установив в них соответствующие значения, а затем установив 1 в **SelectionUpdate**. Когда **SelectionUpdate** установлен в 1, эти элементы записываются в пакет и посылаются на удаленный узел на хранение.

Примечание. При добавлении продуктов для вывода и накопления, все элементы выбора остаются применимы только к текущему накапливаемому продукту.

Имя элемента	Тип DDE	Доступ	Описание
Selection	Целый	ЧЗ	Установка в этот элемент значения номера выборки обновит все элементы выбора соответствующими данными для этой выборки.
SelectionCauseCode	Целый	ЧЗ	Устанавливает код специальной причины для заданной выборки.
SelectionCauseString	Тексто-вый (128)	ТЧ	Выводит описание кода специальной причины для заданной выборки.
SelectionComment	Тексто-вый (50)	ЧЗ	Используется для чтения и записи любых смешанных комментариев для заданной выборки.
SelectionCp	Действи-	ТЧ	Выводит выборочный момент

	тельный		заданной выборки.
SelectionCpk	Действи-тельный	ТЧ	Выводит центральный выборочный момент заданной выборки.
SelectionDate	Тексто-вый (8)	ТЧ	Устанавливает дату для заданной выборки в формате ГГ-ММ-ДД. Если дата введена некорректно, принимается текущая дата.
SelectionFlag	Дискрет-ный	ЧЗ	Устанавливает флаг для заданной выборки.
SelectionIgnoreValue	Дискрет-ный	ЧЗ	Устанавливает режим, при котором заданная выборка будет игнорироваться, если контрольный граф выводится в режиме AutoScaled.
SelectionMx	Действи-тельный	ТЧ	Устанавливает номер индивидуального измерения (x= от 1 до 25), содержащий значение выборки.
SelectionProduct	Текстовый (32)	ТЧ	Выводит имя продукта для заданной выборки.
SelectionRUCL	Действи-тельный	ТЧ	Выводит диапазон верхних пределов управления для заданной выборки.
SelectionRLCL	Действи-тельный	ТЧ	Выводит диапазон нижних пределов управления для заданной выборки.
SelectionR	Действи-тельный	ТЧ	Выводит диапазон для заданной выборки.
SelectionRBAR	Действи-тельный	ТЧ	Выводит среднее для заданной выборки.
SelectionSample	Действи-тельный	ТЧ	Выводит значение в точке заданной выборки.
SelectionSampleBar	Действи-тельный	ТЧ	Выводит среднее для заданной выборки в заданной точке.
SelectionTarget	Действи-тельный	ТЧ	Выводит значение задания для заданной выборки.
SelectionTime	Текстовый (8)	ТЧ	Выводит строку времени для заданной выборки.
SelectionUpdate	Дискрет-ный	ЧЗ	Обновляет изменения в полях выбора (Selection).
SelectionXUSL	Действи-тельный	ТЧ	Устанавливает верхний предел характеристики заданной выборки.
SelectionXLSL	Действи-тельный	ТЧ	Устанавливает нижний предел характеристики заданной выборки.
SelectionXUCL	Действи-тельный	ТЧ	Устанавливает верхний предел управления для заданной выборки.
SelectionXLCL	Действи-тельный	ТЧ	Устанавливает нижний предел управления для заданной выборки.

SelSPC2L3Out2SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "2 из последних 3 выборок вне 2 стандартных отклонений SS".
SelSPC4L5Out1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "4 из последних 5 выборок вне 1 стандартного отклонения SS".
SelSPCConSampAltUpDnMsg	Целый	ТЧ	Тэг сообщения аларма "Последующие выборки варьируются вверх и вниз".
SelSPCConSampln1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки внутри 1 стандартного отклонения".
SelSPCConSamplncDecMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки уменьшаются или увеличиваются".
SelSPCConSampOneSideCLMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки по одну сторону от центральной линии".
SelSPCConSampOut1SDMsg	Текстовый	ТЧ	Тэг сообщения аларма "Последующие выборки вне 1 стандартного отклонения".
SelSPCNLNOutNSDMsg	Текстовый	ТЧ	Тэг сообщения аларма "? из последних ? выборок вне ? стандартных отклонений".
SelSPCNLNOutNSDSSMsg	Текстовый	ТЧ	Тэг сообщения аларма "? из последних ? выборок вне ? стандартного отклонения SS".
SelSPCOutRCtrlMsg	Текстовый	ТЧ	Тэг сообщения аларма графа диапазона "Диапазон вне контрольного предела".
SelSPCOutXCtrlMsg	Текстовый	ТЧ	Тэг сообщения аларма графа X "Выборка вне контрольного предела".
SelSPCOutSpecMsg	Текстовый	ТЧ	Тэг сообщения аларма "Выборка вне предела спецификации".

Примечание. Множественные косвенные наборы данных (Indirect Datasets) могут быть установлены и связаны с одним и тем же набором данных. В этом случае значению Selection каждого косвенного набора данных можно установить разные номера выборок. Это позволит просматривать подробную информацию многих выборок для одного набора данных.

Отладка функций SQL в сценариях

Все SQL функции возвращают код результата *ResultCode*, с помощью которого можно найти и устранить ошибку. Функция **SQLErrorMsg()** возвращает сообщение об ошибке, связанное с кодом результата.

Пример:

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

где: **ErrorMsg** — внутренний текстовый тэг.
ResultCode — целое значение, полученной от предыдущей функции SQL.

Сообщения об ошибках с кодом результата

За информацией по недокументированным кодам результата, пожалуйста, обращайтесь к описанию по вашей конкретной базе данных (а также к главе, приведенной ниже «Коды ошибок конкретных баз данных») и загляните в журнал WWLogger за дополнительной информацией.

Функция **SQLErrorMsg()** устанавливает значение тэга типа сообщение ErrorMsg. Ниже приведен список некоторых возможных кодов результата SQL вместе с соответствующими сообщениями об ошибках и их описание:

Код результата	Сообщение об ошибке	Описание
0	**Не было ошибок**	Команда выполнена успешно.
-5	**Нет больше рядов для выборки**	Достигнута последняя запись в таблице.
-1001	**Недостаточно памяти**	Недостаточно памяти для выполнения этой функции.
-1002	**Неверная связь**	Переданный функции идентификатор ConnectionID имеет недопустимое значение.
-1003	**Не было найдено списка привязок**	Указанное имя Списка Связей (Bind List) не существует.
-1004	**Не были найдены шаблоны**	Имя указанного шаблона таблицы не существует.
-1005	**Внутренняя ошибка**	Возникла внутренняя ошибка. Обратитесь за технической поддержкой!
-1006	**Строка - null**	Предупреждение - строка, считанная из базы данных, нулевая.
-1007	**Строка была усечена**	Предупреждение – строка, считанная из базы данных, длиннее 131 символа и обрезана.
-1008	**Нет оператора where**	Не задан условный оператор Where для выполнения команды Delete.
-1009	**Ошибка при соединении**	Поищите в журнале WWLogger более подробное описание неудавшегося соединения.
-1010	**База данных, указанная в блоке DB= строки соединения, не существует**	Указанной базы данных не существует.
-1011	**Не было выбрано рядов**	Попытка выполнить команды **SQLNumRows(), SQLFirst(),**

		SQLNext() или **SQLPrev()**, не выполнив предварительно функцию **SQLSelect()**.
-4149	**Команда соединения или указатель запроса неверный**	Возможно, некорректно определен тип столбца. Например, если таблица форм определена с помощью столбца с типом **character** вместо **char** для файла dBase, будет возвращена ошибка.

Коды ошибок конкретных баз данных

Oracle

Сообщение об ошибке	Решение
ORA-03112 - Host String Syntax error	Если вы не запустили **NETINIT.EXE**, занесите его в группу автоматического запуска Windows. Если вы запустили **NETINIT.EXE**, и хотите установить более одного соединения или сессии, вам понадобится выделить память. Сделайте это, установив тэг **WIN_REMOTE_SESSIONS** в файле конфигурации **CONFIG.ORA** равным числу требуемых соединений. В приведенном примере будет выделена память для 4 соединений: **WIN_REMOTE_SESSIONS=4**
ORA-3121 - No interface driver connected	Запустите резидентную программу SQL*NET TSR в соответствии с вашей системой сетевого обеспечения до входа в Windows и использования SQL доступа в InTouch.
ORA-6435 - NetBIOS: Unable to add local name to name table	Должно быть запущено сетевое программное обеспечение (Novell, LAN Manager и т. п.)
ORA-09301 - Local kernel only supported in standard mode	Укажите имя сервера в строке соединения («**SRVR=**»).
ORA-06430 - Unable to make connection	Проверьте точность и правильность порядка задания атрибутов в строке соединения.

Sybase или Microsoft SQL Server

Сообщение об ошибке	Решение
You cannot have more than one statement active at a time	Вы пытаетесь выполнить команду SQL после вызова функции **SQLSelect()**. Выполните **SQLEnd()**, чтобы освободить системные ресурсы, занятые **SQLSelect()** или используйте отдельный идентификатор ConnectionID для второго выражения.
There is not enough memory available to process the command	Попробуйте перезапустить рабочую станцию клиента.
Invalid object name table name	Имя таблицы не существует в базе данных, которую вы используете. Попробуйте задать DB=имя базы данных.

dBASE

Сообщение об ошибке	Решение
File or DLL not found	Для Windows файл QEDBF.DLL должен быть либо в текущем каталоге, либо в каталоге \Windows\system\ , указанном в переменной среды *path*.
Invalid connection	Убедитесь в том, что в пути *path* есть соответствующие DLL-файлы. Для работы с файлами DBF потребуется файл QLDBF.DLL.
The connection or statement handle you provided is not valid	Для получения подробной информации загляните в журнал WWLogger. Возможна синтаксическая ошибка в SQL выражении.

Предметный указатель

“$”

$AccessLevel, 1-2
$AlarmLogging, 1-3
$AlarmPrinterError, 1-3
$AlarmPrinterNoPaper, 1-4
$AlarmPrinterOffline, 1-4
$AlarmPrinterOverflow, 1-5
$ApplicationChanged, 1-5
$ApplicationVersion, 1-6
$ChangePassword, 1-6
$ConfigureUsers, 1-7
$Date, 1-7
$DateString, 1-8
$DateTime, 1-8
$Day, 1-8
$HistoricalLogging, 1-9
$Hour, 1-9
$InactivityTimeout, 1-10
$InactivityWarning, 1-10
$LogicRunning, 1-11
$Minute, 1-11
$Month, 1-11
$Msec, 1-12
$NewAlarm, 1-12
$ObjHor, 1-12
$ObjVer, 1-13
$Operator, 1-13
$OperatorEntered, 1-14
$PasswordEntered, 1-14
$Second, 1-15
$StartDdeConversations, 1-15
$System, 1-15
$Time, 1-16
$TimeString, 1-16
$Year, 1-16

“.”

.Ack, 2-7
.Alarm, 2-8
.AlarmDevDeadband, 2-9
.AlarmEnabled, 2-10
.AlarmGroup, 2-59
.AlarmValDeadband, 2-11
.Caption, 2-65
.ChartLength, 2-12“,2-13
.Comment, 2-13“,2-36
.DevTarget, 2-14
.DisplayMode, 2-14
.Enabled, 2-66
.EngUnits, 2-15
.HiHiLimit, 2-15
.HiHiStatus, 2-16
.HiLimit, 2-17
.HiStatus, 2-17
.ListCount, 2-66
.ListIndex, 2-67
.LoLimit, 2-18
.LoLoLimit, 2-18
.LoLoStatus, 2-19
.LoStatus, 2-20
.MajorDevPct, 2-20
.MajorDevStatus, 2-21
.MaxEU, 2-22
.MaxRange, 2-23
.MaxRaw, 2-24
.MinEU, 2-25
.MinorDevPct, 2-26
.MinorDevStatus, 2-27
.MinRange, 2-28
.MinRaw, 2-29
.Name, 2-29
.NewIndex, 2-67
.NextPage, 2-59
.Normal, 2-31
.NumAlarms, 2-60
.OffMsg, 2-32
.OnMsg, 2-32
.PageNum, 2-60
.Pen1-.Pen8, 2-33
.PrevPage, 2-61
.PriFrom, 2-61
.PriTo, 2-62
.ProvidesReg, 2-62
.ProvidesRet, 2-63
.Quality, 2-36
.QualityLimit, 2-39
.QualityLimitString, 2-39
.QualityStatus, 2-40
.QualitySubstatus, 2-41
.QualitySubstatusString, 2-42
.QueryState, 2-63
.QueryType, 2-64
.RawValue, 2-42

.ReadOnly, 2-68
.Reference, 2-43
.ReferenceComplete, 2-43
.ROCPct, 2-44
.ROCStatus, 2-45
.ScooterLockLeft, 2-46
.ScooterLockRight, 2-47
.ScooterPosLeft, 2-48
.ScooterPosRight, 2-49
.SPCStatus, 2-50
.Successful, 2-64
.TagID, 2-50
.TimeDate, 2-51
.TimeDateString, 2-51
.TimeDateTime, 2-51
.TimeDay, 2-52
.TimeHour, 2-52
.TimeMinute, 2-52
.TimeMonth, 2-53
.TimeMsec, 2-53
.TimeSecond, 2-53
.TimeTime, 2-54
.TimeTimeString, 2-54
.TimeYear, 2-54
.TopIndex, 2-68
.TotalPages, 2-65
.Unack, 2-55
.UpdateCount, 2-56
.UpdateInProgress, 2-57
.UpdateTrend, 2-58
.Value, 2-58“,2-69
.Visible, 2-70

“A”

Abs(), 3-2
Ack(), 3-2
ActivateApp(), 3-2
almAckAll(), 3-3
almAckRecent(), 3-3
almAckSelect(), 3-3“,3-5
almDefQuery(), 3-5
almMoveWindow(), 3-6
almQuery(), 3-7
almSelectAll(), 3-7
almSelectItem(), 3-8
almShowStats(), 3-8
ArcCos(), 3-8
ArcSin(), 3-9
ArcTan(), 3-9

“C”

ChangePassword(), 3-10
Cos(), 3-10

“D”

DDE
  Control. *См.* WWControl()
  Execute. *См.* WWExecute()
  Poke. *См.* WWPoke()
  Request. *См.* WWRequest()
DialogStringEntry(), 3-11
DialogValueEntry(), 3-12
DText(), 3-13

“E”

ErrorCode, A-3
ErrorNumber, A-2
Exp(), 3-14

“F”

FileCopy(), 3-14
FileDelete(), 3-15
FileMove(), 3-16
FileReadFields(), 3-18
FileReadMessage(), 3-19
FileWriteFields(), 3-20
FileWriteMessage(), 3-21

“G”

GetNodeName(), 3-21
GetPropertyD(), 3-22
GetPropertyI(), 3-22
GetPropertyM(), 3-23
GOT Функции
  GetPropertyD(), 3-22
  GetPropertyI(), 3-22
  GetPropertyM(), 3-23
  SetPropertyD(), 3-57
  SetPropertyI(), 3-57
  SetPropertyM(), 3-58

“H”

Hide(), 3-23
HideSelf(), 3-23
HTGetLastError(), 3-24
HTGetPenName(), 3-24
HTGetTimeAtScooter(), 3-25

HTGetTimeStringAtScooter(), 3-25
HTGetValue(), 3-26
HTGetValueAtScooter(), 3-27
HTGetValueAtZone(), 3-28
HTScrollLeft(), 3-29
HTScrollRight(), 3-29
HTSelectTag(), 3-30
HTSetPenName(), 3-30
HTUpdateToCurrentTime(), 3-31
HTZoomIn(), 3-31
HTZoomOut(), 3-33

“I”

InfoAppActive(), 3-34
InfoAppTitle(), 3-34
InfoDisk(), 3-35
InfoFile(), 3-36
InfoInTouchAppDir(), 3-37
InfoResources(), 3-37
Int(), 3-38
IOSetAccessName(), 3-39
IOSetItem(), 3-40
IsAnyAsynchFunctionBusy(), 3-41

“L”

Log(), 3-41
LogMessage(), 3-42
LogN(), 3-42

“P”

Pi(), 3-42
PlaySound(), 3-43
PrintHT(), 3-43
PrintWindow(), 3-45

“R”

RecipeDelete(), 3-46
RecipeGetMessage(), 3-47
RecipeGetMessages, A-5
RecipeLoad, A-3
RecipeLoad(), 3-48
RecipeSave(), 3-49
RecipeSelectNextRecipe(), 3-50
RecipeSelectPreviousRecipe(), 3-51
RecipeSelectRecipe(), 3-52
RecipeSelectUnit(), 3-53
RestartWindowViewer(), 3-54
ResultCode, A-16
Round(), 3-54

“S”

SendKeys(), 3-55
SetDDEAppTopic(), 3-56
SetDDEItem(), 3-56
SetPropertyD(), 3-57
SetPropertyI(), 3-57
SetPropertyM(), 3-58
Sgn(), 3-58
Show(), 3-59
ShowAt(), 3-59
ShowHome(), 3-60
ShowTopLeftAt(), 3-60
Sin(), 3-60
SPC
  Имена элементов, A-5
SPC .Fields
  .SPCStatus, 2-50
SPCConnect(), 3-61
SPCDisplayData(), 3-61
SPCLocateScooter(), 3-62
SPCMoveScooter(), 3-62
SPCSaveSample(), 3-62
SPCSelectDataset(), 3-63
SPCSelectProduct(), 3-63
SPCSetControlLimits(), 3-63
SPCSetMeasurement(), 3-64
SPCSetProductCollected(), 3-64
SPCSetProductDisplayed(), 3-64
SPCSetRangeLimits(), 3-65
SPCSetSpecLimits(), 3-65
SQL Функции
  SQLSelect(), 3-83
SQLAppendStatement(), 3-65
SQLClearParam(), 3-67
SQLClearStatement(), 3-67
SQLClearTable(), 3-67
SQLCommit(), 3-68
SQLConnect(), 3-69
SQLCreateTable(), 3-70
SQLDelete(), 3-71
SQLDisconnect(), 3-71
SQLDropTable(), 3-73
SQLEnd(), 3-73
SQLErrorMsg, A-16
SQLErrorMsg(), 3-73
SQLExecute(), 3-75
SQLFirst(), 3-75
SQLGetRecord(), 3-75
SQLInsert(), 3-76

SQLInsertEnd(), 3-76
SQLInsertExecute(), 3-78
SQLInsertPrepare(), 3-78
SQLLast(), 3-78
SQLLoadStatement(), 3-79
SQLManageDSN(), 3-79
SQLNext(), 3-80
SQLNumRows(), 3-80
SQLPrepareStatement(), 3-80
SQLPrev(), 3-82
SQLRollback(), 3-82
SQLSelect(), 3-83
SQLSetParamChar(), 3-86
SQLSetParamDate(), 3-86
SQLSetParamDateTime(), 3-87
SQLSetParamDecimal(), 3-87
SQLSetParamFloat(), 3-88
SQLSetParamInt(), 3-88
SQLSetParamLong(), 3-88
SQLSetParamNull(), 3-89
SQLSetParamTime(), 3-90
SQLSetStatement(), 3-90
SQLTransact(), 3-92
SQLUpdate(), 3-93
SQLUpdateCurrent(), 3-94
Sqrt(), 3-94
StartApp(), 3-94
StringASCII(), 3-95
StringChar(), 3-95
StringFromIntg(), 3-96
StringFromReal(), 3-96
StringFromTime(), 3-97
StringInString(), 3-98
StringLeft(), 3-98
StringLen(), 3-99
StringLower(), 3-99
StringMid(), 3-100
StringReplace(), 3-101
StringRight(), 3-103
StringSpace(), 3-103
StringTest(), 3-104
StringToIntg(), 3-105
StringToReal(), 3-105
StringTrim(), 3-106
StringUpper(), 3-106
System Tags
  $Hour, 1-9
System Type System Tags
  $Hour, 1-9

“T”

Tan(), 3-107
Text(), 3-107
Trunc(), 3-108

“W”

wcAddItem(), 3-108
wcClear(), 3-109
wcDeleteItem(), 3-109
wcDeleteSelection(), 3-110
wcErrorMessage, A-2
wcErrorMessage(), 3-110
wcFindItem(), 3-111
wcGetItem(), 3-111
wcGetItemData(), 3-112
wcInsertItem(), 3-113
wcLoadList(), 3-113
wcLoadText(), 3-115
wcSaveList(), 3-116
wcSaveText(), 3-117
wcSetItemData(), 3-118
WWControl(), 3-119
WWExecute(), 3-120
WWPoke(), 3-121
WWRequest(), 3-122

“A”

Архивные поля
  .ChartLength, 2-12“,2-13
  .DisplayMode, 2-14
  .MaxRange, 2-23
  .MinRange, 2-28
  .Pen1-.Pen8, 2-33
  .ScooterLockLeft, 2-46
  .ScooterLockRight, 2-47
  .ScooterPosLeft, 2-48
  .ScooterPosRight, 2-49
  .TagID, 2-50
  .UpdateCount, 2-56
  .UpdateInProgress, 2-57
  .UpdateTrend, 2-58
Архивные функции
  HTGetLastError(), 3-24
  HTGetPenName(), 3-24
  HTGetTimeAtScooter(), 3-25
  HTGetTimeStringAtScooter(), 3-25
  HTGetValue(), 3-26
  HTGetValueAtScooter(), 3-27
  HTGetValueAtZone(), 3-28

HTScrollLeft(), 3-29
HTScrollRight(), 3-29
HTSetPenName(), 3-30
HTUpdateToCurrentTime(), 3-31
HTZoomIn(), 3-31
HTZoomOut(), 3-33
PrintHT(), 3-43

“В”

Внешние
Типы тэгов, 2-3
Внутренние системные переменные. *См.*
Системные тэги

“И”

Имена элементов
SPC DDE, A-5
Элементы выбора SPC DDE, A-13
Элементы ручного ввода SPC DDE, A-12
Элементы текущей выборки SPC DDE, A-8
Элементы управления и вывода на экран DDE, A-5

“К”

Коды ошибок конкретных баз данных
dBASE, A-19
Oracle, A-18
Sybase или Microsoft SQL Server, A-19

“М”

Математические функции
Abs(), 3-2
ArcCos(), 3-8
ArcSin(), 3-9
ArcTan(), 3-9
Cos(), 3-10
DText(), 3-13
Exp(), 3-14
Int(), 3-38
Log(), 3-41
LogN(), 3-42
Pi(), 3-42
Round(), 3-54
Sgn(), 3-58
Sin(), 3-60
Sqrt(), 3-94
Tan(), 3-107
Trunc(), 3-108

“О”

Отладка, A-1
Рецептурные функции, A-3
Функции SQL, A-16
Отладка рецептурных функций, A-3
Отладка функций SQL, A-16
Отладка функций сценариев, A-1

“П”

Переменная ConnectionID, 3-69
Переменная ConnectString, 3-69
Поля тэгов
.Comment, 2-13“,2-36
.EngUnits, 2-15
.MaxEU, 2-22
.MaxRaw, 2-24
.MinEU, 2-25
.MinRaw, 2-29
.Name, 2-29
.OffMsg, 2-32
.OnMsg, 2-32
.Quality, 2-36
.QualityLimit, 2-39
.QualityLimitString, 2-39
.QualityStatus, 2-40
.QualitySubstatus, 2-41
.QualitySubstatusString, 2-42
.RawValue, 2-42
.Reference, 2-43
.ReferenceComplete, 2-43
.TagID, 2-50
.TimeDate, 2-51
.TimeDateString, 2-51
.TimeDateTime, 2-51
.TimeDay, 2-52
.TimeHour, 2-52
.TimeMinute, 2-52
.TimeMonth, 2-53
.TimeMsec, 2-53
.TimeSecond, 2-53
.TimeTime, 2-54
.TimeTimeString, 2-54
.TimeYear, 2-54
.Value, 2-58
Поля тэгов алармов
.Ack, 2-7
.Alarm, 2-8
.AlarmDevDeadband, 2-9
.AlarmEnabled, 2-10

.AlarmValDeadband, 2-11
.DevTarget, 2-14
.HiHiLimit, 2-15
.HiHiStatus, 2-16
.HiLimit, 2-17
.HiStatus, 2-17
.LoLimit, 2-18
.LoLoLimit, 2-18
.LoLoStatus, 2-19
.LoStatus, 2-20
.MajorDevPct, 2-20
.MajorDevStatus, 2-21
.MinorDevPct, 2-26
.MinorDevStatus, 2-27
.Normal, 2-31
.ROCPct, 2-44
.ROCStatus, 2-45
.Unack, 2-55
Прочие функции
DialogStringEntry(), 3-11
DialogValueEntry(), 3-12
GetPropertyD(), 3-22
GetPropertyI(), 3-22
GetPropertyM(), 3-23
Hide(), 3-23
HideSelf(), 3-23
HTSelectTag(), 3-30
IOSetAccessName(), 3-39
IOSetItem(), 3-40
LogMessage(), 3-42
PlaySound(), 3-43
PrintWindow(), 3-45
SendKeys(), 3-55
SetDDEAppTopic(), 3-56
SetDDEItem(), 3-56
SetPropertyD(), 3-57
SetPropertyI(), 3-57
SetPropertyM(), 3-58
Show(), 3-59
ShowAt(), 3-59
ShowHome(), 3-60
ShowTopLeftAt(), 3-60

“C”

Свойства групп элементов управления Windows
.Caption, 2-65
.Enabled, 2-66
.ListCount, 2-66
.ListIndex, 2-67
.NewIndex, 2-67
.ReadOnly, 2-68
.TopIndex, 2-68
.Value, 2-69
.Visible, 2-70
Свойства группы распределенных алармов
.AlarmGroup, 2-59
.NextPage, 2-59
.NumAlarms, 2-60
.PageNum, 2-60
.PrevPage, 2-61
.PriFrom, 2-61
.PriTo, 2-62
.ProvidesReg, 2-62
.ProvidesRet, 2-63
.QueryState, 2-63
.QueryType, 2-64
.Successful, 2-64
.TotalPages, 2-65
Системные тэги
$AccessLevel, 1-2
$AlarmLogging, 1-3
$AlarmPrinterError, 1-3
$AlarmPrinterNoPaper, 1-4
$AlarmPrinterOffline, 1-4
$AlarmPrinterOverflow, 1-5
$ApplicationChanged, 1-5
$ApplicationVersion, 1-6
$ChangePassword, 1-6
$ConfigureUsers, 1-7
$Date, 1-7
$DateString, 1-8
$DateTime, 1-8
$Day, 1-8
$HistoricalLogging, 1-9
$InactivityTimeout, 1-10
$InactivityWarning, 1-10
$LogicRunning, 1-11
$Minute, 1-11
$Month, 1-11
$Msec, 1-12
$NewAlarm, 1-12
$ObjHor, 1-12
$ObjVer, 1-13
$Operator, 1-13
$OperatorEntered, 1-14
$PasswordEntered, 1-14
$Second, 1-15
$StartDdeConversations, 1-15
$System, 1-15

$Time, 1-16
$TimeString, 1-16
$Year, 1-16
Системные тэги алармов
$AlarmLogging, 1-3
$AlarmPrinterError, 1-3
$AlarmPrinterError, 1-4
$AlarmPrinterOverflow, 1-5
$NewAlarm, 1-12
$System, 1-15
Системные тэги архивирования
$HistoricalLogging, 1-9
Системные тэги доступа
$AccessLevel, 1-2
$ChangePassword, 1-6
$ConfigureUsers, 1-7
$InactivityTimeout, 1-10
$InactivityWarning, 1-10
$Operator, 1-13
$OperatorEntered, 1-14
$PasswordEntered, 1-14
Системные тэги приложения
$ApplicationChanged, 1-5
$ApplicationVersion, 1-6
Системные тэги системного типа
$Date, 1-7
$DateString, 1-8
$DateTime, 1-8
$Day, 1-8
$LogicRunning, 1-11
$Minute, 1-11
$Month, 1-11
$Msec, 1-12
$ObjHor, 1-12
$ObjVer, 1-13
$Second, 1-15
$StartDdeConversations, 1-15
$Time, 1-16
$TimeString, 1-16
$Year, 1-16
Системные функции
ActivateApp(), 3-2
FileCopy(), 3-14
FileDelete(), 3-15
FileMove(), 3-16
FileReadFields(), 3-18
FileReadMessage(), 3-19
FileWriteFields(), 3-20
FileWriteMessage(), 3-21
GetNodeName(), 3-21
InfoAppActive(), 3-34
InfoAppTitle(), 3-34
InfoDisk(), 3-35
InfoFile(), 3-36
InfoInTouchAppDir(), 3-37
InfoResources(), 3-37
IsAnyAsynchFunctionBusy(), 3-41
RestartWindowViewer(), 3-54
StartApp(), 3-94
Системныетэгиалармов
$AlarmPrinterOffline, 1-4
Сообщения об ошибках
Функции распределенных алармов, A-2
Функции элементов управления окна, A-2
Сообщения об ошибках и их описание, A-3
Сообщения об ошибках с кодом результата, A-16
Сообщения с кодами ошибок, A-5
Строковые функции
StringASCII(), 3-95
StringChar(), 3-95
StringFromIntg(), 3-96
StringFromReal(), 3-96
StringFromTime(), 3-97
StringInString(), 3-98
StringLeft(), 3-98
StringLen(), 3-99
StringLower(), 3-99
StringMid(), 3-100
StringReplace(), 3-101
StringRight(), 3-103
StringSpace(), 3-103
StringTest(), 3-104
StringToIntg(), 3-105
StringToReal(), 3-105
StringTrim(), 3-106
StringUpper(), 3-106
Text(), 3-107

“T”

Таблица типов тэгов, 2-4
Типы тэгов, 2-2
I/O, 2-3
Memory, 2-2
Внешние
I/O Discrete, 2-3
I/O Integer, 2-3
I/O Real, 2-3
Внутренние
Memory Discrete, 2-2

Memory Integer, 2-2
Memory Message, 2-2
Memory Real, 2-2
Косвенные, 2-3
Прочие, 2-4
Group Var, 2-4
Hist Trend, 2-4
Indirect Analog, 2-3
Indirect Discrete, 2-3
Indirect Message, 2-3
SuperTags, 2-4
Tag ID, 2-4
Типы тэгов и поля, 2-4

“Ф”

Функции
Abs(), 3-2
Ack(), 3-2
ActivateApp(), 3-2
almAckAll(), 3-3
almAckRecent(), 3-3
almAckSelect(), 3-3“,3-5
almDefQuery(), 3-5
almMoveWindow(), 3-6
almQuery(), 3-7
almSelectAll(), 3-7
almSelectItem(), 3-8
almShowStats(), 3-8
ArcCos(), 3-8
ArcSin(), 3-9
ArcTan(), 3-9
ChangePassword(), 3-10
Cos(), 3-10
DialogStringEntry(), 3-11
DialogValueEntry(), 3-12
DText(), 3-13
Exp(), 3-14
FileCopy(), 3-14
FileDelete(), 3-15
FileMove(), 3-16
FileReadFields(), 3-18
FileReadMessage(), 3-19
FileWriteFields(), 3-20
FileWriteMessage(), 3-21
GetNodeName(), 3-21
GetPropertyD(), 3-22
GetPropertyI(), 3-22
GetPropertyM(), 3-23
Hide(), 3-23
HideSelf(), 3-23
HTGetLastError(), 3-24
HTGetPenName(), 3-24
HTGetTimeAtScooter(), 3-25
HTGetTimeStringAtScooter(), 3-25
HTGetValue(), 3-26
HTGetValueAtScooter(), 3-27
HTGetValueAtZone(), 3-28
HTScrollLeft(), 3-29
HTScrollRight(), 3-29
HTSelectTag(), 3-30
HTSetPenName(), 3-30
HTUpdateToCurrentTime(), 3-31
HTZoomIn(), 3-31
HTZoomOut(), 3-33
InfoAppActive(), 3-34
InfoAppTitle(), 3-34
InfoDisk(), 3-35
InfoFile(), 3-36
InfoInTouchAppDir(), 3-37
InfoResources(), 3-37
Int(), 3-38
IOSetAccessName(), 3-39
IOSetItem(), 3-40
IsAnyAsynchFunctionBusy(), 3-41
Log(), 3-41
LogMessage(), 3-42
LogN(), 3-42
Pi(), 3-42
PlaySound(), 3-43
PrintHT(), 3-43
PrintWindow(), 3-45
RecipeDelete(), 3-46
RecipeGetMessage(), 3-47
RecipeLoad(), 3-48
RecipeSave(), 3-49
RecipeSelectNextRecipe(), 3-50
RecipeSelectPreviousRecipe(), 3-51
RecipeSelectRecipe(), 3-52
RecipeSelectUnit(), 3-53
RestartWindowViewer(), 3-54
Round(), 3-54
SendKeys(), 3-55
SetDDEAppTopic(), 3-56
SetDDEItem(), 3-56
SetPropertyD(), 3-57
SetPropertyI(), 3-57
SetPropertyM(), 3-58
Sgn(), 3-58
Show(), 3-59
ShowAt(), 3-59

ShowHome(), 3-60
ShowTopLeftAt(), 3-60
Sin(), 3-60
SPCConnect(), 3-61
SPCDisplayData(), 3-61
SPCLocateScooter(), 3-62
SPCMoveScooter(), 3-62
SPCSaveSample(), 3-62
SPCSelectDataset(), 3-63
SPCSelectProduct(), 3-63
SPCSetControlLimits(), 3-63
SPCSetMeasurement(), 3-64
SPCSetProductCollected(), 3-64
SPCSetProductDisplayed(), 3-64
SPCSetRangeLimits(), 3-65
SPCSetSpecLimits(), 3-65
SQLAppendStatement(), 3-65
SQLClearParam(), 3-67
SQLClearStatement(), 3-67
SQLClearTable(), 3-67
SQLCommit(), 3-68
SQLConnect(), 3-69
SQLCreateTable(), 3-70
SQLDelete(), 3-71
SQLDisconnect(), 3-71
SQLDropTable(), 3-73
SQLEnd(), 3-73
SQLErrorMsg(), 3-73
SQLExecute(), 3-75
SQLFirst(), 3-75
SQLGetRecord(), 3-75
SQLInsert(), 3-76
SQLInsertEnd(), 3-76
SQLInsertExecute(), 3-78
SQLInsertPrepare(), 3-78
SQLLast(), 3-78
SQLLoadStatement(), 3-79
SQLManageDSN(), 3-79
SQLNext(), 3-80
SQLNumRows(), 3-80
SQLPrepareStatement(), 3-80
SQLPrev(), 3-82
SQLRollback(), 3-82
SQLSelect(), 3-83
SQLSetParamChar(), 3-86
SQLSetParamDate(), 3-86
SQLSetParamDateTime(), 3-87
SQLSetParamDecimal(), 3-87
SQLSetParamFloat(), 3-88
SQLSetParamInt(), 3-88
SQLSetParamLong(), 3-88
SQLSetParamNull(), 3-89
SQLSetParamStatement(), 3-90
SQLSetParamTime(), 3-90
SQLTransact(), 3-92
SQLUpdate(), 3-93
SQLUpdateCurrent(), 3-94
Sqrt(), 3-94
StartApp(), 3-94
StringASCII(), 3-95
StringChar(), 3-95
StringFromIntg(), 3-96
StringFromReal(), 3-96
StringFromTime(), 3-97
StringInString(), 3-98
StringLeft(), 3-98
StringLen(), 3-99
StringLower(), 3-99
StringMid(), 3-100
StringReplace(), 3-101
StringRight(), 3-103
StringSpace(), 3-103
StringTest(), 3-104
StringToIntg(), 3-105
StringToReal(), 3-105
StringTrim(), 3-106
StringUpper(), 3-106
Tan(), 3-107
Text(), 3-107
Trunc(), 3-108
wcAddItem(), 3-108
wcClear(), 3-109
wcDeleteItem(), 3-109
wcDeleteSelection(), 3-110
wcErrorMessage(), 3-110
wcFindItem(), 3-111
wcGetItem(), 3-111
wcGetItemData(), 3-112
wcInsertItem(), 3-113
wcLoadList(), 3-113
wcLoadText(), 3-115
wcSaveList(), 3-116
wcSaveText(), 3-117
wcSetItemData(), 3-118
WWControl(), 3-119
WWExecute(), 3-120
WWPoke(), 3-121
WWRequest(), 3-122
Отладка, A-1
Функции Quick-сценариев. *См.* Функции

Функции SPC
SPCConnect(), 3-61
SPCDisplayData(), 3-61
SPCLocateScooter(), 3-62
SPCMoveScooter(), 3-62
SPCSaveSample(), 3-62
SPCSelectDataset(), 3-63
SPCSelectProduct(), 3-63
SPCSetControlLimits(), 3-63
SPCSetMeasurement(), 3-64
SPCSetProductCollected(), 3-64
SPCSetProductDisplayed(), 3-64
SPCSetRangeLimits(), 3-65
SPCSetSpecLimits(), 3-65
Функции SQL
SQLAppendStatement(), 3-65
SQLClearParam(), 3-67
SQLClearStatement(), 3-67
SQLClearTable(), 3-67
SQLCommit(), 3-68
SQLConnect(), 3-69
SQLCreateTable(), 3-70
SQLDelete(), 3-71
SQLDisconnect(), 3-71
SQLDropTable(), 3-73
SQLEnd(), 3-73
SQLErrorMsg(), 3-73
SQLExecute(), 3-75
SQLFirst(), 3-75
SQLGetRecord(), 3-75
SQLInsert(), 3-76
SQLInsertEnd(), 3-76
SQLInsertExecute(), 3-78
SQLInsertPrepare(), 3-78
SQLLast(), 3-78
SQLLoadStatement(), 3-79
SQLManageDSN(), 3-79
SQLNext(), 3-80
SQLNumRows(), 3-80
SQLPrepareStatement(), 3-80
SQLPrev(), 3-82
SQLRollback(), 3-82
SQLSetParamChar(), 3-86
SQLSetParamDate(), 3-86
SQLSetParamDateTime(), 3-87
SQLSetParamDecimal(), 3-87
SQLSetParamFloat(), 3-88
SQLSetParamInt(), 3-88
SQLSetParamLong(), 3-88
SQLSetParamNull(), 3-89
SQLSetParamStatement(), 3-90
SQLSetParamTime(), 3-90
SQLTransact(), 3-92
SQLUpdate(), 3-93
SQLUpdateCurrent(), 3-94
Функции WWDDE
WWControl(), 3-119
WWExecute(), 3-120
WWPoke(), 3-121
WWRequest(), 3-122
Функции алармов
Ack(), 3-2
Функции доступа
ChangePassword(), 3-10
Функции распределенных алармов
almAckAll(), 3-3
almAckRecent(), 3-3
almAckSelect(), 3-3“,3-5
almDefQuery(), 3-5
almMoveWindow(), 3-6
almQuery(), 3-7
almSelectAll(), 3-7
almSelectItem(), 3-8
almShowStats(), 3-8
Сообщения об ошибках, A-2
Функции рецептов
RecipeDelete(), 3-46
RecipeGetMessage(), 3-47
RecipeLoad(), 3-48
RecipeSave(), 3-49
RecipeSelectNextRecipe(), 3-50
RecipeSelectPreviousRecipe(), 3-51
RecipeSelectRecipe(), 3-52
RecipeSelectUnit(), 3-53
Функции элементов управления Windows
wcAddItem(), 3-108
wcClear(), 3-109
wcDeleteItem(), 3-109
wcDeleteSelection(), 3-110
wcErrorMessage(), 3-110
wcFindItem(), 3-111“,3-112
wcGetItem(), 3-111
wcInsertItem(), 3-113
wcLoadList(), 3-113
wcLoadText(), 3-115
wcSaveList(), 3-116
wcSaveText(), 3-117
wcSetItemData(), 3-118
Функции элементов управления окна
Сообщения об ошибках, A-2