Ай Ти Ви групп

Программный комплекс

«Интеллект»

Руководство по программированию

Версия 2.1. 2

Москва 2011

ITV ИНТЕЛЛЕКТ. ТЕХНОЛОГИИ. ВИДЕО.

axxon Video Surveillance and Security Solutions

Содержание

1 Введение

1.1 Назначение программного комплекса «Интеллект»

Программный комплекс «Интеллект» предназначен для создания промышленных масштабируемых гибко настраиваемых (адаптируемых) интегрированных систем безопасности на основе цифровых систем видеонаблюдения и аудиоконтроля.

Программный комплекс «Интеллект» обладает следующими основополагающими функциональными возможностями:

1. Интеграция цифровых систем видеонаблюдения и аудиоконтроля со смежными информационными системами, различного типа охранным оборудованием, вспомогательным программным обеспечением сторонних производителей при использовании интегрированных открытых интерфейсов информационного взаимодействия.
2. Совместимость с широким перечнем охранного оборудования и информационных систем безопасности, в частности, таких, как охранно-пожарная сигнализация, системы контроля доступа, видеокамеры, информационные системы анализа, распознавания и идентификации объектов (событий) на видеоизображении.
3. Централизованная регистрация и обработка событий, генерация оповещений и управляющих воздействий в соответствии с гибко настраиваемыми алгоритмами.
4. Практически неограниченные возможности масштабирования, адаптации к специфике решаемых задач, перераспределения используемых ресурсов при изменении количества или качества задач по мониторингу состояния подконтрольных объектов и управления различного рода оборудованием.

1.2 Настройка логических взаимосвязей между объектами в программном комплексе «Интеллект»

Функциональные возможности программного комплекса «Интеллект» основаны на логическом взаимодействии между объектами. Общие сведения о способах настройки логических взаимосвязей приведены в таблице (см. Таб. 1.2-1).

Таб. 1.2-1. Способы настройки логических взаимосвязей

Способ настройки логической взаимосвязи	**Описание**	**Реализация**	**Пример**
Панели настройки объектов системы	Базовая настройка взаимодействия между объектами системы	Реализуется с использованием функционала объектов системы – см. документ «Программный комплекс Интеллект: Руководство администратора»	Настройка отображения видеосигнала в интерфейсном окне «Монитор»
Макрокоманда	Настройка простых взаимосвязей между объектами, функционал которых не позволяет выполнить требуемые	Реализуется с использованием функционала объекта «Макрокоманда» – см. документ «Программный комплекс Интеллект: Руководство администратора»	Настройка включения исполнительного устройства (реле) при замыкании луча

Способ настройки логической взаимосвязи	Описание	Реализация	Пример
	операции		
Программа	Настройка комплексных взаимосвязей между объектами, если функционал объекта «Макрокоманда» не позволяет выполнить требуемые операции	Реализуется на базе объекта «Программа» в виде кода на встроенном в ПК «Интеллект» языке программирования – см. настоящее Руководство	Требуется каждые 15 минут возвращать поворотные камеры в исходное положение и делать снимок
Скрипт		Реализуется на базе объекта «Скрипт» в виде кода на языке JavaScript – см. документ «Программный комплекс Интеллект: Руководство по программированию (JavaScript)»	

1.3 Назначение и структура руководства

Документ «Программный комплекс «Интеллект». Руководство по программированию» является справочно-информационным пособием по программированию на встроенном языке ПК «Интеллект» и предназначен для системных администраторов, специалистов по установке и настройке, пользователей с правами администрирования цифровых систем видеонаблюдения и аудиоконтроля, созданных на основе программного комплекса «Интеллект».

Программирование в ПК «Интеллект» позволяет автоматизировать управление системой путем настройки комплексных логических взаимосвязей между объектами.

В данном Руководстве представлены следующие материалы:

1. инструментарий программирования;
2. описание синтаксиса встроенного языка программирования;
3. примеры программ на встроенном языке.

2 Инструментарий программирования

2.1 Системный объект «Программа»

Системный объект «Программа» предназначен для инициализации в программе «Интеллект» программы, разработанной на языке JavaScript (или на собственном языке программирования ПК «Интеллект»), и задания параметров его выполнения.

Системный объект «Программа» создается на базе объекта «Программы» на вкладке «Программирование» диалогового окна «Настройка системы» (Рис. 2.1-1).

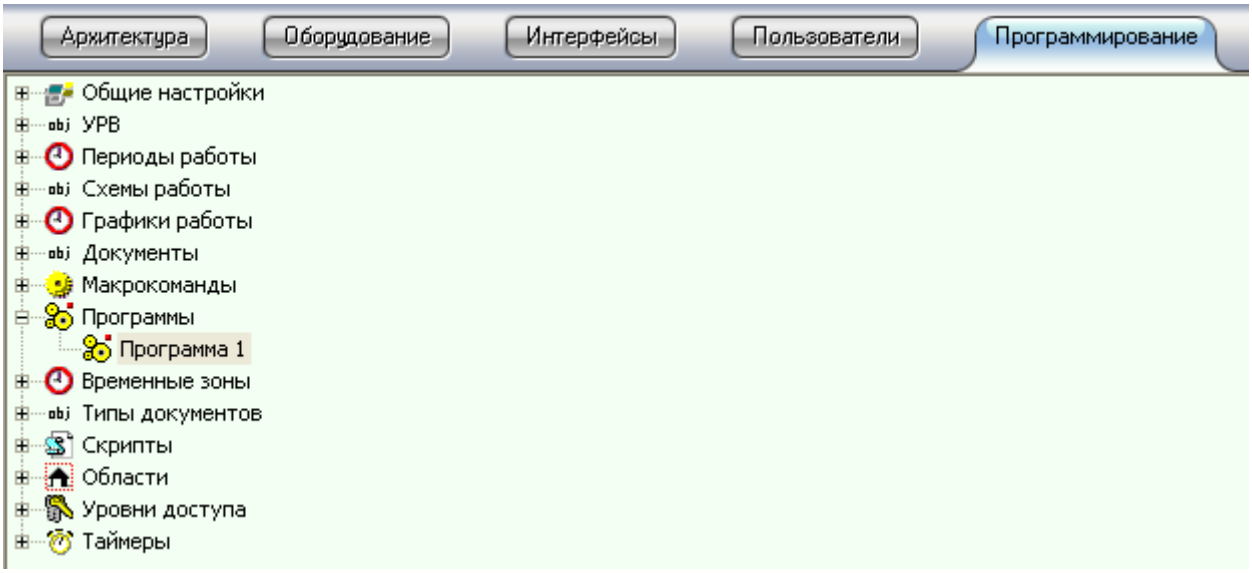


Рис. 2.1-1 Создание Объект «Программа»

Панель настройки системного объекта «Программа» представлена на Рис. 2.1-2.

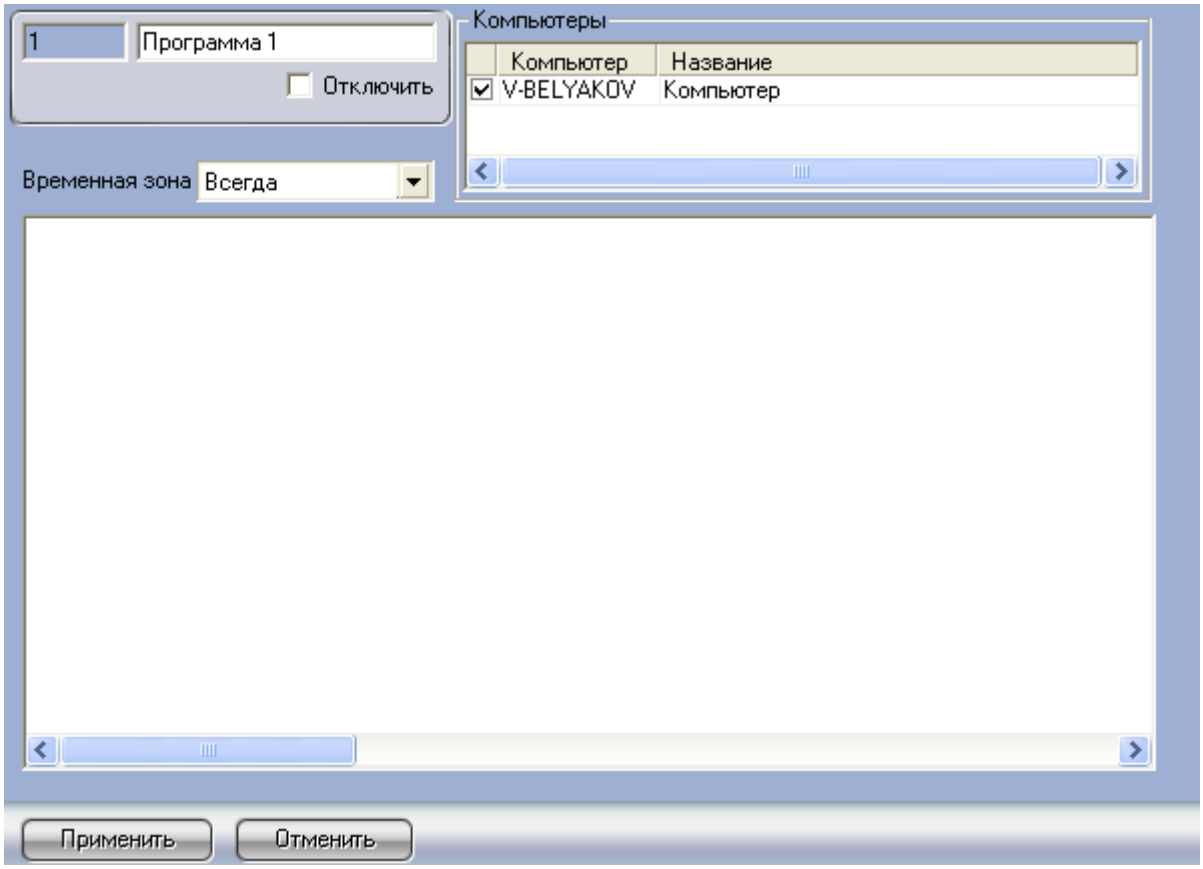


Рис. 2.1-2 Панель настройки системного объекта «Программа»

В панели настройки системного объекта «Программа» указываются временная зона выполнения программы и компьютеры (ядра), на которых требуется выполнять программу.

На панели настройки системного объекта «Программа» размещен текстовой редактор для написания и редактирования кода программы.

В текстовом редакторе на панели настроек системного объекта «Программа» есть возможность отмены действия и повтора с помощью горячих клавиш. Для отмены действия нажмите «Alt+Backspace», для повтора – «Ctrl+Y».

2.2 Отладочное окно

В программном комплексе "Интеллект" существует возможность в реальном времени просматривать все события и реакции, происходящие в системе. События и реакции со свойствами объектов отображаются в Отладочном окне, откуда их можно скопировать в буфер обмена Windows для последующего использования в программах.

2.2.1 Включение Отладочного окна

Отладочное окно по умолчанию выключено. Для включения Отладочного окна необходимо выполнить следующую последовательность действий:

1. Завершить работу программного комплекса «Интеллект».
2. Запустить утилиту «Расширенная настройка» Tweaki.exe.

Примечание. Отладочное окно можно включить и без использования утилиты tweaki.exe. Для этого следует установить строковый параметр «Debug» равным 1, 2, либо 3 в разделе «HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect» реестра ОС Windows.

3. Выбрать раздел «Intellect» в дереве, расположенном в левой части диалогового окна утилиты.
4. Изменить значение параметра «Debug mode» с «None» на «Debug 1», «Debug 2» или «Debug 3».
5. Нажать кнопку «Ок».
6. Запустить программный комплекс «Интеллект».
7. В Главной панели управления программы «Интеллект» появится новый пункт «Отладочное окно» (см. Рис. 2.2-1).

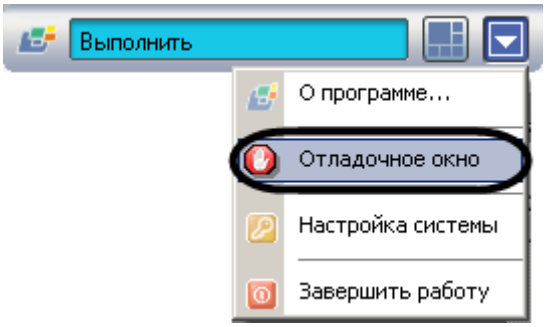


Рис. 2.2-1. Появление пункта «Отладочное окно» в Главной панели управления

8. Выбрать пункт «Отладочное окно» в Главной панели управления для отображения Отладочного окна на экране монитора (см. Рис. 2.2-1). Выбранный пункт меню «Отладочное окно» будет отмечен флажком (см. Рис. 2.2-2).

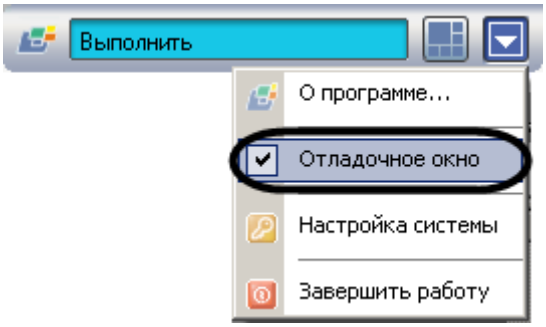


Рис. 2.2-2. Вид пункта «Отладочное окно» в случае его выбора

Для того чтобы скрыть Отладочное окно, требуется повторно выбрать пункт «Отладочное окно» в Главной панели управления.

Примечание. Для выключения Отладочного окна следует выбрать значение «None» для параметра «Debug mode» в утилите tweaki.exe, либо установить строковый параметр «Debug» равным 0 в разделе «HKEY_LOCAL_MACHINE\ SOFTWARE\ ITV\ Intellect» реестра ОС Windows. Данные операции проводятся при выгруженном ПК «Интеллект».

2.2.2 Работа с Отладочным окном

Внешний вид Отладочного окна представлен на Рис. 2.2-3. В Отладочном окне выводится последовательность событий и реакций в системе.

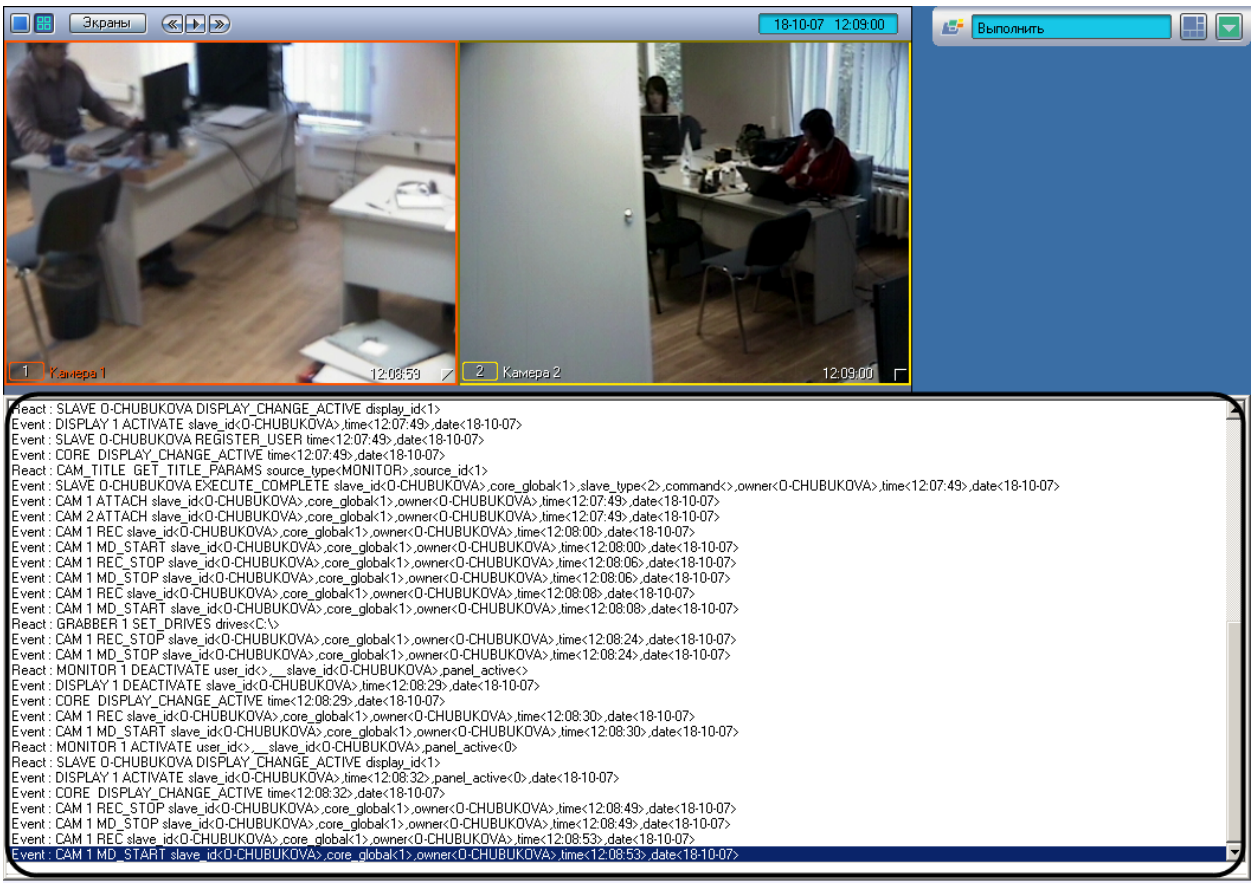


Рис. 2.2-3. Отладочное окно

Отладочное окно обладает следующими свойствами:

1. располагается поверх других окон;
2. для изменения размеров Отладочного окна следует использовать мышь;
3. существует возможность копировать информацию о событии или реакции в буфер обмена Windows для последующего использования в программах.

Чтобы прочитать и/или скопировать в буфер обмена Windows информацию о событии или реакции, необходимо выполнить следующую последовательность действий:

1. Выделить требуемую строку в Отладочном окне.
2. Щелкнуть правой кнопкой мыши по выделенной строке. В результате выполнения операции появится окно «Message», содержащее информацию о требуемом событии или реакции (см. Рис. 2.2-4).
3. Для копирования в буфер обмена Windows следует выделить требуемые сведения, после чего нажать «Ctrl+C»

   Примечание. Для операций с текстом в окне «Message» удобно использовать контекстное меню (вызывается щелчком правой кнопкой мыши по выделенному тексту).
4. Для закрытия окна «Message» необходимо нажать « ».

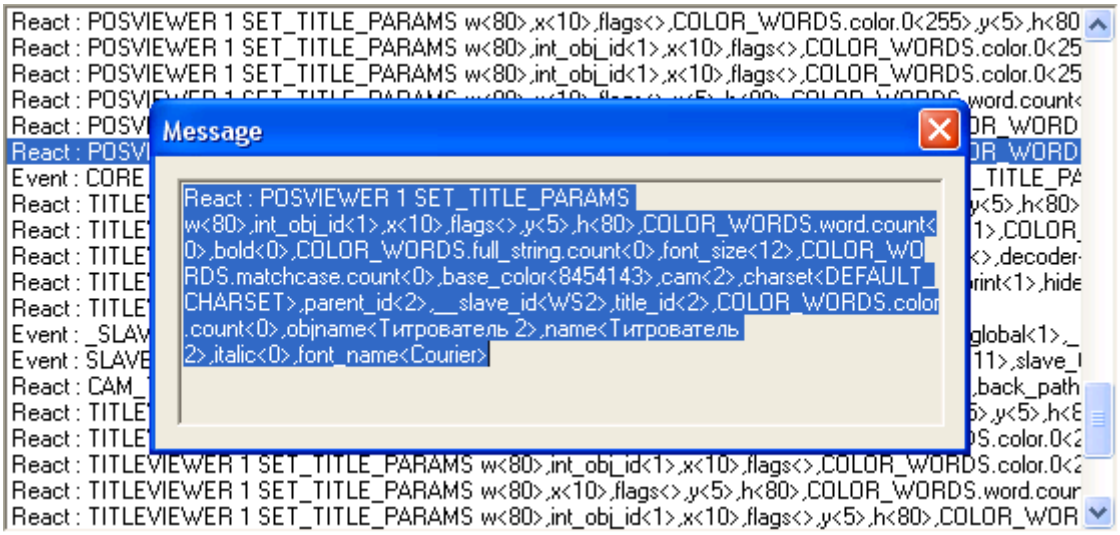


Рис. 2.2-4. Копирование сообщения для использования в программе

Копирование информации о событии или реакции в буфер обмена Windows завершено.

2.3 Синтаксический анализатор

Встроенный синтаксический анализатор позволяет отслеживать правильность написания основных зарегистрированных слов, таких как OnEvent, DoReact, OnTime, Wait, Sleep и др. Эти зарегистрированные слова отмечаются черным цветом в поле текста программы. Следует отметить, что за правильностью написания параметров команд анализатор не следит, и нужно быть особенно внимательным в этих случаях.

```
OnEvent("MACRO","2","RUN")
{[
fn="D:\Intellect\Bmp\Person\1.bmp";

DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<"+fn+">");
DoReact("DIALOG","operator","CLOSE_ALL");
Sleep(500);
DoReact("DIALOG","operator","RUN");
]}

OnEvent("MACRO","3","RUN")
{[
fn="D:\Intellect\Bmp\Person\1.bmp";

DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<"+fn+">");
```

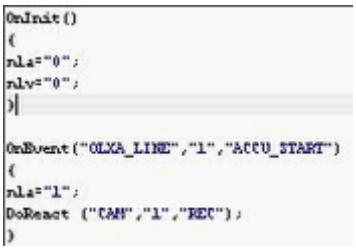
Для изменения размера шрифта используйте сочетания клавиш

CTRL и «+» для увеличения шрифта

```
OnInit()
{
n1a="0";
n1v="0";
}

OnEvent("OLXA_LINE","1","ACCU_START")
{
n1a="1";
DoReact ("CAM","1","REC");
}
```

CTRL и «-» для уменьшения шрифта



2.4 Рекомендуемый порядок написания программ

1. Постановка общей задачи.
2. Разбитие задачи на подзадачи.
3. Написание подзадач и их отладка.
4. Поиск и исправление ошибок.

2.4.1 Постановка общей задачи

Нужно четко представлять, что должно происходить в системе при определенных событиях. Определить ID устройств, участвующих в генерации событий и действий.

2.4.2 Разбитие задачи на подзадачи

Если задача подразумевает обработку нескольких различных событий, то имеет смысл четко представить действия системы на каждое их этих событий. По возможности нужно исключить возможность бесконечного зацикливания выполнения скриптов, т.е исключить всяческие рекурсивные действия, если конечно они не предусматривают выполнение поставленной задачи.

2.4.3 Написание подзадач и их отладка

Наиболее сложным в написании скриптов является написание списка действий с возможным использованием логических и циклических операций. По опыту эта часть программирования наиболее долго отлаживается. Зачастую генерация события, требующая обработки, является не очень удобной, тем более на реальном объекте, например срабатывание пожарного датчика или движение по камере, достаточно удаленной от места программирования - от сервера с Ядром системы. В этом случае рекомендуется на этапе отладки действий генерировать событие вручную, самое удобное - это запуск пустой макрокоманды. После отладки тела скрипта в событие вместо запуска пустой макрокоманды подставляется реальное событие. Кроме того можно проверить и наоборот - убедиться в правильности написания реального события не запуская списка действий

можно вставив вместо списка действий - запуск пустой макрокоманды и посмотреть ее выполнение в отладочном окне.

2.4.4 Поиск и исправление ошибок

Встроенный синтаксический анализатор на этапе запуска программы проверяет правильность написания названий функций, но не проверяет правильность расстановки ключевых символов - запятых, точек с запятой, вложенности скобок. Поэтому ошибки, если они есть, будут проявляться только на этапе исполнения тела программы.

3 Описание синтаксиса

Скрипт состоит из набора процедур.

Все операторы, выполняемые внутри процедур, формируются в блоки {..}.

Если нужно вставить комментарий, то перед комментарием требуется поставить спецсимволы //.

3.1 Описание переменных

Все переменные, используемые в системе – строковые.

Для сравнения строковых переменных и значений используется функция: boolstrequal (строка1,строка2). Функция "boolstrequal" возвращает значение, отличное от нуля, если строки равны (см. раздел «Описание функций»).

Для произведения целочисленных действий используется функция: str(строка1) (см. раздел «Описание функций»).

3.2 Описание процедур

3.2.1 Стандартные процедуры

Существуют 3 стандартные процедуры, которые могут быть выполнены при возникновении соответствующего события:

1. OnInit() **–** используется для инициализации переменных (задания первоначальных значений), которые будут в дальнейшем использоваться при выполнении скриптов. Выполняется до старта всех модулей системы. Рекомендуется использовать один вызов процедуры на все существующие скрипты.

Пример использования:

```
OnInit()
{
        flag=1;
        num=8;  //на старте системы будут проинициализированы переменные
}
```

2. OnTime (день недели (1-7), день-месяц-год, часы, минуты, секунды) – Запуск в определенный момент времени.

```
OnTime(W,D,X,Y,H,C,S)
{
        //W - день недели (0 - понедельник, 6 - воскресенье);
        //D - дата в формате "число-месяц-год", 16 августа 2001 года это "16-08-01"
        //X,Y - зарезервировано
        //H - час
        //C - минуты
        //S - секунды
```

```
        // ВЫПОЛНЯЯ СРАВНЕНИЕ С ПАРАМЕТРАМИ, ДАЛЕЕ УКАЗЫВАЕТСЯ ДЕЙСТВИЕ
}
```

Примеры использования:

```
OnTime(W,"16-08-01",X,Y,"11","11","30")
{
        // помещенный здесь код сработает 16 августа 2001 года в 11 часов 11 минут 30 секунд
}

OnTime(W,D,X,Y,"11","11","30")
{
        // помещенный здесь код сработает каждый день в 11 часов 11 минут 30 секунд
}

OnTime(W,"16-08-01",X,Y,H,C,S)
{
        // помещенный здесь код, будет срабатывать 16 августа 2001 года
        // каждую секунду
}

OnTime(W,"16-08-01",X,Y,"11","11",S)
{
        // помещенный здесь код ,будет срабатывать 16 августа 2001 года
        // с 11 часов 11 минут по 11 часов 12 минут каждую секунду
}

OnTime("0",D,X,Y,"21","0","0")
{
        // помещенный здесь код ,будет срабатывать каждый понедельник
        // в 21 часов 00 минут 00 секунд
}
```

3. OnEvent(тип источника, номер, событие) – запуск по определенному событию от объекта системы. Основная процедура при написании скриптов.

Примеры использования:

```
OnEvent("GRAY","1","ON")
```

```
{
        // Выполнится при замыкании луча №1
}


OnEvent("CAM","12","MD_START")
{
        // Выполнится при срабатывании детектора движения камеры №12
}
```

Каждая процедура, имеющая параметры, может встречаться в коде много раз с различными параметрами. При возникновении события система выполнит те из них, параметры которого совпадут с параметрами возникшего события.

Параметр процедуры может быть определенным или нет. В первом случае его значение берется в кавычки, в последнем случае параметр обозначается латинскими буквами, и процедура будет выполнена для всех событий, для которых его можно определить.

Примеры использования:

```
OnEvent("GRAY","1","ON")       // Выполнится при замыкании луча №1
{
        i=1;
        i=i+1;    //т.к. переменные строковые, то сумма будет равна «11»
        j=1;
        j=str(j+1);          // str - это функция преобразования числа к строке. Внутри функции str вначале происходит конвертация всех строковых переменных (в случае их наличия) в целочисленные, затем происходит сложение чисел, следовательно сумма будет равна «2»
}


OnEvent("GRAY",N,"ON")  // Выполнится при замыкании любого луча
{
        if(strequal(N,"3")
        {
          // выполнится если это луч 3
        }
}
```

3.2.2 Создание собственных процедур

Все собственные процедуры, описанные в скрипте, должны находиться в том же теле программы и перед процедурами, в которых они вызываются.

```
        procedure ProcedureName(список параметров)
```

```
{
//тело процедуры
}
```

Внимание! Имена параметров должны состоять из одного символа, в верхнем регистре.

Примеры использования:

```
procedure ProcedureName(A,B)
{
    n=A+" "+B;
    //при запуске макроса 1 n=«Макрокоманда 1», при запуске макроса 16 n=«Макрокоманда 16»
}

OnEvent("MACRO",N,"RUN")
{
    a1=N;
    a2="Макрокоманда";
    ProcedureName(a2,a1);
}
```

3.3 Описание операторов

Список операторов используемых для описания действий:

1. DoReact (тип объекта, номер, действие[,параметры]) – выполнить действие

Пример использования:

```
OnEvent("GRAY","1","ON")
{
    DoReact("GRELE","1","ON");    //при замыкании луча 1 замкнуть реле1
}
```

2. DoCommand(командная строка) – запуск командной строчки

Пример использования:

```
OnEvent("GRAY","1","ON")
{
    DoCommand("notepad.exe");   //при замыкании луча 1 запустить «Блокнот»
}
```

3. Wait(кол-во секунд) - ждать N секунд;

Sleep(кол-во миллисекунд) - ждать N миллисекунд.

Операторы ожидания должны быть выделены в отдельный поток. Отдельный поток выделяется квадратными скобками.

Пример**.** При замыкании Луча №1 Реле №1 будет замыкаться на 5 секунд.

```
OnEvent("GRAY","1","ON")
{
	[
	DoReact("GRELE","1","ON");
	Wait(5);
	DoReact("GRELE","1","OFF");
	]
}
```

4. Функция проверки состояния объекта:

CheckState(тип объекта, номер, состояние) – результат будет равен 1, если состояние объекта соответствует действительности, иначе 0.

В качестве параметров могут быть выражения. Константные значения берутся в кавычки.

Пример. При замыкании луча №1 проверяется состояние камеры №2 и если состояние «Тревога», то замкнуть реле №1

```
OnEvent("GRAY","1","ON")
{
	if(CheckState("CAM","2","ALARMED"))
	{
		DoReact("GRELE","1","ON");
	}
}
```

5. Условный оператор:

```
	If (выражение)
	{
	… // если результат выражения не 0
	}
	else
	{
	… // если результат выражения равен 0
	}
```

Часть оператора else {} может отсутствовать.

Пример использования:

```
OnEvent ("MACRO","1","RUN")
{
        x=5;
        if(x>10) {y=2;}   // если "x" больше чем 10, то y=2
        else {y=3;}       //иначе y=3
}
```

6. Оператор цикла:

```
        For (выражение 1; выражение 2; выражение 3)
        {
        …
        }
```

Выражение 1 выполнится в начале цикла, пока выражение 2 истинно, будет выполняться тело цикла, после каждого выполнения тела цикла будет выполняться выражение 3.

Пример. При замыкании луча №1 реле №1 будет замыкаться и размыкаться с интервалом в 1 секунду и это будет происходить 10 раз.

```
OnEvent ("GRAY","1","ON")
{
        [
        for(i=0;i<10;i=str(i+1))
        {
          DoReact("GRELE","1","ON");
          Wait(1);
          DoReact("GRELE","1","OFF");
          Wait(1);
        }
        ]
}
```

7. DoReactGlobal (тип объекта, номер, состояние) – функция, генерирующая реакции системных объектов. При этом генерируемая реакция рассылается по всем ядрам системы, соединенным по сети.

Пример. При выполнении макрокоманды №1 ставить камеру №1 на охрану.

```
OnEvent("MACRO","1","RUN")
{
```

DoReactGlobal("CAM","1","ARM");

}

8. NotifyEventGlobal (тип объекта, номер, состояние) – функция, генерирующая системные события. При этом генерируемые события рассылаются по всем ядрам системы, соединенным по сети.

Пример. При выполнении макрокоманды №1 генерировать событие «Запись на диск» для камеры №1. Команду отправлять по всем ядрам системы в виде события для регистрации в Протоколе событий.

OnEvent("MACRO","1","RUN")

{

NotifyEventGlobal ("CAM","1","REC");

}

Примечание. Если нет необходимости в рассылки события по всем ядрам системы, можно воспользоваться функцией NotifyEvent.

3.4 Операции и выражения

В таблице (см. Таб. 3.4-1) представлены общее описание и примеры использования операций сравнения, арифметических и условных операций.

Таб. 3.4-1. Использование операций

Оператор	**Общее описание, пример использования**
Операции сравнения	
>	Оператор сравнения – больше. Пример смотрите в «if» и «for»
<	Оператор сравнения – больше. Пример смотрите в «if» и «for»
Арифметические операции	

+	Операция сложение. Пример использования:
	OnEvent ("MACRO","1","RUN")
	{
	x=5;
	y=10;
	i=x+y; // складывает как строковые т.е. 5+10=510
	e=str(x+y); // складывает как числа 5+10=15
	}

-	Операция вычитание. Пример использования:

Оператор	Общее описание, пример использования

	OnEvent ("MACRO","1","RUN")
	{
	x=5;
	y=10;
	i=x-y; // вычитание как числа 5-10=-5
	e=str(x-y); // вычитание как числа 5-10=-5
	}

*	Умножение. Пример использования:
	OnEvent ("MACRO","1","RUN")
	{
	x=5;
	y=10;
	i=x*y; // умножает как числа 5*10=50
	e=str(x*y); // умножает как числа 5*10=50
	}

/	Деление. Пример использования:
	OnEvent ("MACRO","1","RUN")
	{
	x=5;
	y=10;
	i=x/y; // делит как числа 5/10=0.5
	e=str(x/y); // делит как числа 5/10=0.5
	}

%	Остаток от целочисленного деления. Пример использования.
	OnEvent ("MACRO","1","RUN")
	{
	a=1120.0;
	b=100;
	e=a%b; // остаток от целочисленного деления.. т.е. 1100 делится на 100 а 20 это остаток.
	// если делится без остатка то результат = 0
	}

( )	Группа арифметических операций. Пример использования.
	OnEvent ("MACRO","1","RUN")
	{
	x=100/((5*8)/1.028);
	}

Логические операции	

Оператор	Общее описание, пример использования

&&	Оператор логическое И. Пример использования:
	OnEvent ("MACRO","1","RUN")
	{
	a=1;
	b=2;
	z=3;
	if((a<b)&&(b<z)) {y=1; //если ложь , то else
	}
	else {x=0;}
	}

!	Оператор логического отрицания. Пример использования:
	OnEvent ("CAM",N,"MD_START")
	{
	if(!(strequal(N,”1”,)))
	{
	DoReact(“GRELE”,”1”,””ON)
	}
	else
	{
	DoReact(“GRELE”,”2”,””ON)
	}
	}

3.5 Описание функций

Общее описание и примеры использования математических функций, функций преобразования, форматирования и строковых функций показаны в таблице (см. Таб. 3.5-1).

Таб. 3.5-1. Описание функций

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
МАТЕМАТИЧЕСКИЕ	
sin[1]	Тригонометрическая функция расчета синуса угла. Формат: y=sin(x); где: y - значение функции, x – аргумент функции (в радианах) Пример: y=sin(1.6) Полученное событие: Event : CORE VAR_CHANGED nt_obj_id<1>,value<0.997495>,name<y>,time<15:26:41>,date<21-

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	09-04>
cos[1]	Тригонометрическая функция расчета косинуса угла. Формат: y=cos(x); где: y - значение функции, x – аргумент функции (в радианах) Пример: y=cos(2.2) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.588501>,name<y>,time<16:00:45>,date<21-09-04>
tan[1]	Тригонометрическая функция, возвращает тангенс угла. Формат: y=tan(x); где: y - значение функции, x – аргумент функции (в радианах) Пример: y=tan(1) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.557408>,name<y>,time<16:43:45>,date<21-09-04>
asin[1]	Возвращает арксинус заданного числового выражения. Формат: y=asin(x); где: y-значение функции (в радианах), x-аргумент Пример: y=asin(0.5) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.523599>,name<y>,time<16:46:39>,date<21-09-04>
acos[1]	Возвращает арккосинус заданного числового выражения. Формат: y=acos(x); где: y-значение функции (в радианах), x-аргумент Пример: y=acos(0.55) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.988432>,name<y>,time<16:46:39>,date<21-09-04>
atan[1]	Возвращает арктангенс заданного числового выражения. Формат: y=atan(x); где: y-значение функции (в радианах), x-аргумент Пример: y=atan(1.2) Полученное событие: Event : Event : CORE VAR_CHANGED int_obj_id<1>,value<0.876058>,name<y>,time<17:07:09>,date<21-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
sinh[1]	Функция sinh возвращает гиперболический синус значения аргумента. Формат: y=sinh(x); где: y - значение функции, x – аргумент функции Пример: y=sinh(0.8) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.888106>,name<y>,time<17:12:26>,date<21-09-04>
cosh[1]	Функция cosh возвращает гиперболический косинус значения аргумента. Формат: y=cosh(x); где: y - значение функции, x – аргумент функции Пример: y=cosh(0.35) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.336376>,name<y>,time<17:25:25>,date<21-09-04>
tanh[1]	Тригонометрическая функция расчета угла. Формат: y=tanh(x); где: y - значение функции, x – аргумент функции Пример: y=tanh(0.35) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.419068>,name<y>,time<17:25:25>,date<21-09-04>
exp[1]	Возвращает значение функции e^x, где x - заданное числовое выражение. Формат: y=exp(x); где: y-значение функции, x-аргумент Пример: y=exp(1.65) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<5.20698>,name<y>,time<17:39:22>,date<21-09-04>
log[1]	Возвращает натуральный логарифм (по основанию «e») заданного числового выражения. Формат: y=log(x); где: y-значение функции, x-аргумент Пример: y=log(0.65) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.430783>,name<y>,time<17:43:22>,date<21-09-04>
log10[1]	Возвращает десятичный логарифм (по основанию 10) заданного числового выражения.

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	Формат: y=log10(x); где: y-значение функции, x-аргумент Пример: y=log10(0.05) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-1.30103>,name<y>,time<17:46:28>,date<21-09-04>
sqrt[1]	Возвращает квадратный корень из заданного числового выражения. Формат: y=sqrt(x); где: y-значение функции, x-аргумент Пример: y=sqrt(9) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<3>,name<y>,time<17:25:25>,date<21-09-04>
abs[1]	Функция abs возвращает абсолютное значение целого аргумента Формат: y=abs(x); где: y-значение функции, x-аргумент. Пример: y= abs(-1) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<y>,time<13:39:37>,date<22-09-04>
deg[1]	Тригонометрическая функция расчета угла. Возвращает градусную меру Формат: y=deg(x); где: y – значение функции в градусах, x – значение аргумента в радианах. Пример: y=deg(3.14) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<179.908748>,name<y>,time<13:13:51>,date<22-09-04>
rad[1]	Тригонометрическая функция расчета угла. Формат: y=rad(x); где: y – значение функции в радианах, x – значение аргумента в градусах. Пример: y=rad(180) Полученное событие: Event : CORE VAR_CHANGED value<3.141593>,name<y>,time<15:04:17>,date<17-03-08>
ПРЕОБРАЗОВАНИЕ	
floor[1]	Функция преобразования до целого числа в меньшую сторону. Формат: x= floor(y); где: x-значение функции, y- дробное или целое число. Пример:

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	x= floor(5.55) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<5>,name<x>,time<20:51:48>,date<21-09-04>
ceil[1]	Функция преобразования до целого числа в большую сторону. Формат: x= ceil (y); где: x-значение функции, y-дробное или целое число. Пример: x= ceil(5.55) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<x>,time<20:51:48>,date<21-09-04>

str[1]	Функция преобразования числа к строке.
	Формат: x=str(y); где: x-значение функции, y-аргумент
	Пример:
	z=(9);
	a=str(z);
	b=sqrt(a);
	Полученные события:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<z>,time<14:27:31>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<a>,time<14:27:31>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<3>,name<b>,time<14:27:31>,date<22-09-04>

atof[1]	Функция преобразования строки в число. Формат: x=atof(y); где: x-значение функции, y-аргумент Пример: x="0"; x=str(atof(x)+10); Полученное событие: Event : CORE VAR_CHANGED value<0>,name<x>,time<15:34:44>,date<17-03-08> Event : CORE VAR_CHANGED value<10>,name<x>,time<15:34:44>,date<17-03-08>
val[1]	Функция преобразования строки в число. Формат: x=val(y); где: x-значение функции, y-аргумент Пример: x="10"; x=str(val(x)+2); Полученное событие: Event : CORE VAR_CHANGED value<10>,name<x>,time<15:34:44>,date<17-03-08>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	Event : CORE VAR_CHANGED value<12>,name<x>,time<15:34:44>,date<17-03-08>

int[1]	Преобразование дробного числа в целое (отбросить дробную часть)
	Формат: x=int(y); где: x- значение функции, y- аргумент (дробное число для преобразования)
	Пример:
	y=(2.33);
	x=int(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<2.33>,name<y>,time<16:05:28>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<2>,name<x>,time<16:05:28>,date<22-09-04>

long2time[1]	Используется для преобразования заданного кол-ва секунд во время. Формат: x=long2time(y); где: x- значение функции(время), y- число в секундах Формат исходной записи (аргумента): <ММ> Формат полученной записи: <ЧЧ:ММ:СС> Пример: x=long2time(12345); Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<03:25:45>,name<x>,time<13:53:02>,date<20-09-04>.

time2long[1]	Преобразовать время в кол-во секунд
	Формат: x=time2 long(y); где: x- значение в секундах, y- время в формате <часы>.<минуты>.
	Пример:
	y=(0.15);
	x=time2long(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<0.15>,name<y>,time<19:39:49>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<900>,name<x>,time<19:39:49>,date<22-09-04>

scalar2date[1]	Преобразовать кол-во дней в дату. (Кол-во дней исчисляется с начала нашей эры)
	Формат: x= scalar2date (y); где: x-значение(дата), y-кол-во дней.
	Пример:
	y=(731500);
	x=scalar2date(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<731500>,name<y>,time<19:57:46>,date<22-

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	09-04> Event : CORE VAR_CHANGED int_obj_id<1>,value<12-10-03>,name<x>,time<19:57:46>,date<22-09-04>
scalar[1]	Преобразовать дату в кол-во дней. (Кол-во дней исчисляется с начала нашей эры.) Формат: x=scalar(y); где: x- числовое значение (в сутках), y- дата. Формат записи: <ДД.ММ.ГГГГ> Пример: x=scalar(“19.10.2004”) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<10>,value<731873>,owner<WS1>,name<x>,time<15:24:11>,guid_pk<{42E93AF5-4862-485E-AEF6-D14C7BF79C5B}>,date<08-12-09>

convert_num[1]	Преобразовать число в строку написания этого числа
	
	Формат: x=convert_num(y); где: x- строковое значение числа, y- преобразуемое число.
	
	Пример:
	
	y=(24009921);
	
	x=convert_num(y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<24009921>,name<y>,time<12:37:20>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Двадцать четыре миллиона девять тысяч девятсот двадцать один >,name<x>,time<12:37:20>,date<23-09-04>

convert_cur[1]	Преобразовать число (денежную сумму) в строку написания этого числа и добавить руб. и коп.
	
	Формат: x=convert_cur(y); где: x- строковое значение денежной суммы, y- число(денежная сумма).
	
	Формат записи: <РР.КК>
	
	Пример:
	
	y=(17999.98);
	
	x=convert_cur(y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.98>,name<y>,time<12:49:30>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Семнадцать тысяч девятсот девяносто девять рублей 98 копеек >,name<x>,time<12:49:30>,date<23-09-04>

ФОРМАТИРОВАНИЯ	
number_frm[2]	Форматирование числа

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

	Формат: x=number_frm(y,z); где: x-значение функции, y-исходное число, z- количество цифр после запятой.
	
	Пример:
	
	y=(17999.09998);
	
	x=number_frm(y,3);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.09998>,name<y>,time<14:21:24>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.100>,name<x>,time<14:21:24>,date<23-09-04>

int_frm[2]	Форматирование числа
	
	Формат: x=int_frm(y,z); где: x-значение, y-исходное число, z- количество цифр в числе на выходе.
	
	Пример:
	
	y=(17999.99);
	
	x=int_frm(y,10);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.99>,name<y>,time<14:31:46>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<0000017999>,name<x>,time<14:31:46>,date<23-09-04>

currency_std[1]	Форматирование значения числа представляющего деньги (замена '.' на '-') Формат: x=currency_std(y); где: x- значение функции с измененным форматом, y-число(денежная сумма). Пример: x=currency_std(3.62); Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<3-62>,name<x>,time<13:40:01>,date<23-09-04>
IsVarExist[1]	Функция проверки заданного параметра в событии. Формат: y=IsVarExist(“x”); где: y - значение, x – параметр Если параметр существует, возвращается «1», иначе «0». Пример: p=IsVarExist(“param0”) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<10>,value<0>,owner<WS1>,name<p>,time<12:02:11>,guid_pk<{6A8B5BC9-919C-

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	4098-844A-FBF78FA20820}>,date<14-12-09>
GetObjectIdByParam [3]	Функция, возвращающая первый найденный идентификатор объекта по заданному значению параметра. Id=GetObjectIdByParam ("x","y","z"); где id - возвращаемое значение, x – тип объекта, y – параметр, z –значение параметра. Пример: Id=GetObjectIdByParam("CAM","color","0"); Полученное событие: Event : CORE VAR_CHANGED date<28-02-11>,value<2>,int_obj_id<1>,fraction<218>,name<Id>,guid_pk<{F903A28C-3243-E011-901F-6CF049E58698}>,time<15:02:04>,owner<D-IVANOV> * Id=2 (см. value<2>), если функция возвращает пустое значение (value< >), необходимо проверить правильность написания параметров и самой функции.
СТРОКОВЫЕ	

strequal[2]	Сравнение строк
	
	Формат: x= strequal(z,y); где: x-значение, z и y-сравниваемые строки.
	
	Пример:
	
	z=str(1019);
	
	y=str(1019);
	
	x=strequal(z,y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1019>,name<z>,time<16:51:45>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1019>,name<y>,time<16:51:45>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<x>,time<16:51:45>,date<23-09-04>
	
	* «value<1>» (см. пример выше) - в полученном событии мы получаем либо «value<>» - это означает что сравниваемые строки не совпадают, либо «value<1>» - это значить что сравниваемые строки полностью идентичны друг другу.

strsub[2]	Определение наличия подстроки в строке.
	
	Формат: x=strsub(y,z); где: x-значение, y – строка, в которой ведется поиск, z-подстрока.
	
	Пример №1:
	
	z=str(888123);
	
	y=str(123);
	
	x=strsub(z,y);

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<888123>,name<z>,time<16:07:07>,date<23-09-04> Event : CORE VAR_CHANGED int_obj_id<1>,value<123>,name<y>,time<16:07:07>,date<23-09-04> Event : CORE VAR_CHANGED int_obj_id<1>,value<4>,name<x>,time<16:04:34>,date<23-09-04> Пример №2: z="67hb8vc56"; y="vc"; x=strsub(z,y); Полученное событие: Event : CORE VAR_CHANGED value<67hb8vc56>,name<z>,time<12:15:09>,date<18-03-08> Event : CORE VAR_CHANGED value<vc>,name<y>,time<12:15:09>,date<18-03-08> Event : CORE VAR_CHANGED value<6>,name<x>,time<12:15:09>,date<18-03-08> * "value<4>" (см. пример № 1) - означает индекс в исходной строке, начиная с которого обнаружено первое вхождение подстроки в эту строку. При отрицательном результате поиска функция возвращает значение value<>.
strempty[1]	Определение пуста ли строка. Формат: x=strempty(y); где: x- значение(равно 1 если строка пуста), y-строка. Пример: y=""; x=strempty(y); Полученное событие: Event : CORE VAR_CHANGED value< >, name<y>,time<12:27:32>,date<18-03-08> Event : CORE VAR_CHANGED value<1>,name<x>,time<12:27:32>,date<18-03-08> * значение функции value <> означает, что строка не пуста.

straleft[2]	Выравнивание влево
	
	Формат: x=straleft(y,z); где: x-выровненная строка, y-строка, z-значение выравнивания.
	
	Пример:
	
	y=str(123456789);
	
	x=straleft(y,5);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<123456789>,name<y>,time<18:04:05>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<12345>,name<x>,time<18:04:05>,date<23-09-

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	04> *Примечание. В случае, если число z больше, чем количество символов в строке, функция дополняет исходную строку пробелами справа до тех пор, пока ее длина не станет равна числу z.*

strmid[3]	Взять подстроку
	
	Формат: x=strmid(y,z,w); где: x-строковое значение, y-строка, z- с какой позиции строки, w-длинна подстроки.
	
	Пример:
	
	z=(7);//с какой позиции
	
	w=(9);//длинна
	
	x=strmid("взять подстроку (1 - строка, 2 - с какой позиции, 3 - длинна)",z,w);
	
	y=strmid("взять подстроку (1 - строка, 2 - с какой позиции, 3 - длинна)",17,10);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<z>,time<14:18:08>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<w>,time<14:18:08>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку>,name<x>,time<14:18:08>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1 – строка>,name<y>,time<14:18:08>,date<24-09-04>

strletf[2]	Взять левую часть строки
	
	Формат: y=strleft(s,w); где: y- строковое значение, s-строка, w-длинна(с начала строки)
	
	Пример:
	
	w=(5);//длина
	
	s=("Взять левую часть строки");//строка
	
	y=strleft(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<5>,name<w>,time<14:54:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять левую часть строки>,name<s>,time<14:54:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять>,name<y>,time<14:54:31>,date<24-09-04>

strright[2]	Взять правую часть строки (1 - строка, 2 - длина)
	
	Формат: y=strleft(s,w); где: y- строковое значение, s-строка, w-длинна (с конца строки)
	
	Пример:
	
	w=(6);//длина
	
	s=("Взять правую часть строки");//строка

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
	y=strright(s,w); Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:10:36>,date<24-09-04> Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять правую часть строки>,name<s>,time<15:10:36>,date<24-09-04> Event : CORE VAR_CHANGED int_obj_id<1>,value<строки>,name<y>,time<15:10:36>,date<24-09-04>

strnleft[2]	Взять без левой части строки.
	
	Формат: y=strnleft(s,w); где: y- строковое значение, s-строка, w- длина левой части которая будет отсечена.
	
	Пример:
	
	w=(6);//длина
	
	s=("взять без левой части строки ");//строка
	
	y=strnleft(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:32:38>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без левой части строки >,name<s>,time<15:32:38>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<без левой части строки>,name<y>,time<15:32:38>,date<24-09-04>

srtnright[2]	Взять без правой части строки.
	
	Формат: y=strnright(s,w); где: y- строковое значение, s-строка, w- длина правой части которая будет отсечена.
	
	Пример:
	
	w=(6);//длина
	
	s=("взять без правой части строки");//строка
	
	y=strnright(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:44:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без правой части строки>,name<s>,time<15:44:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без правой части >,name<y>,time<15:44:31>,date<24-09-04>

get_substr[3]	Взять подстроку (1 - строка, 2 - подстрока с которой начать, 3 – подстрока которой завершить, "\r" - конец строки) Формат: y=get_substr(s,w,x); где: y- значение(подстрока), s-строка, w- подстрока с которой начать, x- подстрока которой завершить("\r" - конец строки)

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

	Формат записи: <NN.NN>
	
	Пример:
	
	s=("взять подстроку 1234567890");//строка
	
	w=("по");//подстрока с которой начать
	
	x=("\r");//подстрока которой завершить, "\r" - конец строки
	
	y=get_substr(s,w,x);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять подстроку 1234567890>,name<s>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<по>,name<w>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<\r>,name<x>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку 1234567890>,name<y>,time<16:34:13>,date<24-09-04>
	
	Пример:
	
	s=("взять подстроку 1234567890");//строка
	
	w=("по");//подстрока с которой начать
	
	x=(1);//подстрока которой завершить, "\r" - конец строки
	
	y=get_substr(s,w,x);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять подстроку 1234567890>,name<s>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<по>,name<w>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<x>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку >,name<y>,time<16:36:26>,date<24-09-04>

strltrim[1]	Убрать пробелы слева
	
	Формат: y=strltrim(w); где: y- полученное строковое значение, w- строка.
	
	Пример:
	
	w=(" убрать пробелы слева");//строка
	
	y=strltrim(w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value< убрать пробелы слева>,name<w>,time<17:07:49>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<убрать пробелы слева>,name<y>,time<17:07:49>,date<24-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

strrtrim[1]	Убрать пробелы справа
	
	Формат: y=strrtrim(w); где: y- полученное строковое значение, w- строка.
	
	Пример:
	
	w=("Убрать пробелы справа ");//строка
	
	y=strrtrim(w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Убрать пробелы справа >,name<w>,time<17:18:35>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Убрать пробелы справа>,name<y>,time<17:18:35>,date<24-09-04>

stratrim[1]	Убрать пробелы с обеих сторон
	
	Формат: y=stratrim(w); где: y- полученное строковое значение, w-строка.
	
	Пример:
	
	w=(" убрать пробелы с обеих сторон ");//строка
	
	y=stratrim(w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value< убрать пробелы с обеих сторон >,name<w>,time<17:27:44>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<убрать пробелы с обеих сторон>,name<y>,time<17:27:44>,date<24-09-04>

Примечание. Функции date<ДД-ММ-ГГ > и time<ЧЧ:ММ:СС> возвращают текущие дату и врем я соответственно. Функция pi<3,14159265358979323846264338327 95> возвращает значение числа π.

3.6 Примеры скриптов

Для наглядности и непосредственного закрепления написания скриптов ниже приведены примеры, которые помогут лучше разобраться в способах создания скриптов в системе.

Пример 1. Выводить активную камеру на аналоговый монитор.

Реализация**:**

```
OnEvent ("MONITOR","1","ACTIVATE_CAM")
 {
        DoReact ("CAM",cam,"MUX1");
}
```

Пример 2. Запускать и останавливать патрулирование поворотника по макрокомандам.

Реализация:

```
OnEvent("MACRO","1","RUN")
{
	DoReact("TELEMETRY","1.1","PATROL_PLAY","tel_prior<1>");
}
 OnEvent("MACRO","2","RUN")
{
	DoReact("TELEMETRY","1.1","STOP","tel_prior<1>");
}
```

Пример 3. Выводить тревожную камеру в режим однократора.

Реализация:

```
OnEvent ("CAM",N,"MD_START")
{
	DoReact ("MONITOR","1","ACTIVATE_CAM","cam<"+N+">");
	DoReact ("MONITOR","1","KEY_PRESSED","key<SCREEN.1>");
}
```

Пример 4. Пример бесконечного цикла и выхода из него. Старт цикла по макрокоманде №1, остановка по макрокоманде №2.

Реализация:

```
OnEvent("MACRO","1","RUN") //при запуске макрокоманды №1
{
//квадратные скобки нужны для выделения оператора ожидания в отдельный поток
	 [
	flag=1;
	for(a=1;flag<2;a=1) //оператор цикла
	{
	Sleep(500); //оператор ожидания создает паузу в 500 миллисекунд
	ff="!!!!!!!!!!!!!!!!!";
```

```
        }
        ]
}
OnEvent("MACRO","2","RUN"); //при запуске макрокоманды №2
{
        flag=2;
}
```

Пример 5. Тревожный монитор, на котором всегда остается видео от последней тревожной камеры.

Реализация:

```
OnInit()
{
        counter=0;
}

OnEvent("CAM",T,"MD_START")
{
        if(strequal(counter,"0"))
        {
        DoReact("MONITOR","2","REMOVE_ALL");
        DoReact("MONITOR","2","ADD_SHOW","cam<"+T+">");
        }
        counter=str(counter+1);
}

OnEvent("CAM",M,"MD_STOP")
{
        counter=str(counter-1);
        if(strequal(counter,"0"))
        {
        DoReact("MONITOR","2","ADD_SHOW","cam<"+M+">");
```

```
        }
}
```

Пример 6. Проигрывание звукового файла от прихода одного события, до прихода другого события. (В данном случае это запуск макрокоманд).

Внимание!!! Звуковой файл должен длиться не больше количества секунд, которое указано в операторе Wait.

Реализация:

```
OnEvent("MACRO","1","RUN")
{
        flag=1;
        [
        for(i=1;flag;i=1)
        {
 DoReact("PLAYER","1","PLAY_WAV","file<C:\ Program Files\ Intellect\Wav\cam_alarm_1.wav>");
        Wait(3);
        }
        ]
}

OnEvent("MACRO","8","RUN")
{
        flag=0;
}
```

Пример 7. Есть 2 камеры с поворотными устройствами. Каждые 15 минут нужно повернуть камеры в пресет №1 (предустановка №1) и сделать скриншот. Имя файла – текущее время.

Реализация:

```
 OnTime(W,D,X,Y,H,M,S)
{
if(strequal(M,"0"))
        {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотник 1.1
```

```
        name="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name);
        //Камера 2 Поворотник 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
        }
if(strequal(M,"15"))
        {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотник 1.1
        name="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name);
        //Камера 2 Поворотник 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
        }
if(strequal(M,"30"))
        {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотник 1.1
        name="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name);
        //Камера 2 Поворотник 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
        }
if(strequal(M,"45"))
        {
```

```
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотник 1.1
        name="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name);
        //Камера 2 Поворотник 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
        }
}
```

Пример 8. Микрофон (OLXA_LINE) пишется не синхронно с камерой. По умолчанию микрофон не стоит на охране. Необходимо писать звук как по аккустопуску, так и по детекции от камеры.

Примечание. Команды RECORD_START, RECORD_STOP для микрофона добавлены с версии 4.7.0

На сработку аккустопуска (ACCU_START) и детектора движения (MD_START) включается принудительная запись звука и увеличивается на единицу переменная flag. При окончании аккустопуска и детекции движения переменная flag уменьшается на единицу и запись звука останавливается, только если она равна нулю, т.е. нет ни аккустопуска, ни движения.

Реализация:

```
OnInit()
{
        flag=0;
}

OnEvent("MACRO","1","RUN")
{
        DoReact("PERSON","214","SETUP","facility_code<111>");
}

OnEvent("CAM","3","MD_START")
{
        flag=str(flag+1);
        DoReact("OLXA_LINE","1","RECORD_START");
```

```
}

OnEvent("OLXA_LINE","1","ACCU_START")
{
        flag=str(flag+1);
        DoReact("OLXA_LINE","1","RECORD_START");
}

OnEvent("OLXA_LINE","1","ACCU_STOP")
{
        flag=str(flag-1);
        if (!(flag))
        {
        DoReact("OLXA_LINE","1","RECORD_STOP");
        }
}

OnEvent("CAM","3","MD_STOP")
{
        flag=str(flag-1);
        if (!(flag))
        {
        DoReact("OLXA_LINE","1","RECORD_STOP");
        }
}
```

Пример 9. Есть определенное количество камер (num). Необходимо проверить работу детектора движения по всем камерам (можно использовать для проверки работоспособности датчиков охраны).

Для решения задачи используется эмуляция линейного символьного массива (строка), т.е. заполняется массив символов (у нас это символ «N»). Далее при сработке детектора движения по камере – меняется соответствующий (идентификатору камеры) элемент массива (меняется на "Y"). Таким образом, на выходе у нас символьный массив из «N» (камера не сработала) и «Y» (камера сработала). Подсчитывается количество сработок и выдается сообщение об общем количестве камер и количество камер, у которых сработал детектор. Старт проверки по Макрокоманде №1. Остановка по Макрокоманде №2.

Реализация:

```
OnInit()
{
	run=0;
}
OnEvent("MACRO","1","RUN")
{
	run=1; flag=""; num=8;
	for(i=1;i<str(num+1);i=str(i+1))
	{
	DoReact("CAM",i,"DISARM");
	DoReact("CAM",i,"REC_STOP");
	DoReact("CAM",i,"ARM");
	flag=flag+"N";
	if(i<num) {flag=flag+"|";}
	}
}

OnEvent("CAM",N,"MD_START")
{
	if(run)
	{
	nn=str((N*2)-1);
	flag=strleft(flag,str(nn-1))+"Y"+strright(flag,str(((num*2)-1)-nn));
	}
}

OnEvent("MACRO","2","RUN")
{
	run=0; fin=0;
	for(i=1;i<str(num+1);i=str(i+1))
	{
	tmp=extract_substr(flag,"|",str(i-1));
	if(strequal(tmp,"Y")) {fin=str(fin+1);}
```

```
        DoReact("CAM",i,"DISARM");
        }
        tmp="Всего:"+str(num)+" Сработало:"+str(fin);
        rez=MessageBox("",tmp,0);
}
```

Пример 10. Осуществить патрулирование нескольких зон видимости с помощью пресетов поворотной камеры, с возможностью включения детектора движения на определенных областях этих зон.

Камера №1. 5 зон детектора, 5 предустановок (пресетов). Два этих параметра задаются переменной n. Макрокоманда №1 - старт алгоритма. Макрокоманда №2 - остановка алгоритма. Flag - внутренняя переменная.

При старте алгоритма камера становится в 1-й пресет и ставит на охрану 1-ю зону детектора. Между этими командами задержка 200 миллисекунд, чтобы камера успела встать в пресет. Далее через 5 секунд 1-я зона снимается с охраны и цикл начинается заново но уже с второй зоной и 2-м пресетом. И так далее пока не переберутся все n зон и пресетов. После начинается заново с 1-го. Алгоритм останавливается, если переменная flag обнуляется (с помощью макрокоманды №2).

```
 OnEvent("MACRO","1","RUN")
{
        flag=1;
        n=5;
        [
        for(i=1;flag;i=str(i+1))
        {
                 DoReact("TELEMETRY","1.1.","GO_PRESET","preset<"+i+">,tel_prior<3>");
                Sleep(200);
                DoReact("CAM_ZONE","1."+i,"ARM");
                Wait(5);
                DoReact("CAM_ZONE","1."+i,"DISARM");
                if(strequal(i,n)) {i=0;}
        }
        ]
}
OnEvent("MACRO","2","RUN")
```

```
{
        flag=0;
}
```

3.7 Описание реакций объектов системы

В данной главе указаны все реакции для основных объектов системы.

Примечание. События для объектов системы можно просмотреть одним из следующих способов:

1. *Просмотр содержимого файла intellect.ddi посредством утилиты «ddi.exe» (см. документ «Программный комплекс «Интеллект» видеонаблюдение и аудиоконтроль. Руководство Администратора»).*
2. *Просмотр событий для выбранного объекта системы посредством панели настроек системного объекта «Макрокоманда» (см. документ «Программный комплекс «Интеллект» видеонаблюдение и аудиоконтроль. Руководство Администратора»).*

3.7.1 GRABBER

Объект «Grabber» соответствует системному объекту «Плата видеоввода».

От объекта «Grabber» поступают события, представленные в таблице (см. Таб. 3.7-1). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для платы видеоввода:

```
OnEvent("GRABBER","_id_","_событие_");
```

Таб. 3.7-1. Описание событий от объекта «Grabber»

События	Описание события
"+12V"	Ошибка напряжения +12V.
"+3.3V"	Ошибка напряжения +3.3V.
"+5V"	Ошибка напряжения +5V.
"-12V"	Ошибка напряжения -12V.
"-5V"	Ошибка напряжения -5V.
"CPU_FAN"	Количество оборотов вентилятора.
"CPU_TEMP"	Температура процессора.
"SYS_TEMP"	Температура чипсета MB.
"UPS_COMMLOST"	Потеря связи.
"UPS_FATAL_ERROR"	Ошибка подключения.
"UPS_LOWBATT"	Села батарея.
"UPS_ONBATT"	Переход на питание от батареи.
"UPS_ONLINE"	Восстановление питания от сети.
"UPS_REPLACEBATT"	Требуется замена батареи.
"UPS_SHUTTING"	Выключение.
"VCORE"	Напряжение ядра процессора.

Формат оператора для описания действий с платой видеоввода:

DoReact("GRABBER","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «GRABBER» представлен в таблице (см. Таб. 3.7-2).

Таб. 3.7-2. Список команд и параметров для объекта «Grabber»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" - устанавливает параметры платы видеоввода.	chan<>	Номер PCI слота (0,1,2,…,32).
	mode<>	Скорость граббера/оцифровки (0 – максимальная, 1 – средняя, 2 – минимальная).
	resolution<>	Разрешение (0– стандартное, четверть кадра (384x288); 1 – высокое, полукадр (768x288); 2 – максимальное, кадр (768x576)).
	format<>	Формат видеосигнала (PAL, NTSC).
	drives<>	Диски для записи видеоархива (DRIVE1:\, DRIVE2:\ … DRIVEN:\).
	cams<>	Количество подключенных видеокамер.
	auth<>	
	ip<>	IP-адрес сетевой платы видеоввода.
	name<>	Имя объекта.
	flags <>	Флаги.
	ip_port<>	IP-порт.
	password<>	Пароль.
	type<>	Тип оцифровки.
	username<>	Логин.
	watchdog<>	Включение WatchDog (0 – выключен, 1 – включен).
"SET_DRIVES" - устанавливает диски для записи видеоархива.	drives<>	Диски для записи видеоархива.
"MUX1_OFF" – отключить вывод видео через аналоговый выход 1.	-	-
"MUX2_OFF" - отключить вывод видео через аналоговый выход 2.	-	-
"MUX3_OFF" - отключить вывод видео через аналоговый выход 3.	-	-

Свойства объекта «GRABBER» показаны в таблице (см. Таб. 3.7-3).

Таб. 3.7-3. Свойства объекта «Grabber»

Свойства объекта « GRABBER »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Номер платы видеоввода.

Примеры использования событий и реакций объекта «Плата видеоввода»:

1. Необходимо установить для первой платы видеоввода первый канал, максимальную скорость оцифровки, разрешение – полукадр и формат PAL, при запуске первой макрокоманды.

OnEvent("MACRO","1","RUN"); // запуск макрокоманды 1

{

DoReact("GRABBER","1", "SETUP", "chan<1>,mode<0>,resolution<1>,format<PAL>"); //установка для первой платы видеоввода канал – 1, скорость оцифровки – максимальную, разрешение – полукадр, формат – PAL

}

Примечание. Описание объекта "MACRO" указано ниже (см. раздел «MACRO»).

2. Необходимо при запуске третей макрокоманды установить диски D:\ и F:\ для записи видеоархива.

OnEvent("MACRO","3","RUN"); //запуск макрокоманды 3

{

DoReact("GRABBER","1","SET_DRIVES","drives<D:\,F:\>"); //запись видеоархива на диски D:\ и F:\

}

3. Необходимо вывести первую видеокамеру на первый аналоговый выход платы и отключить первые аналоговые выходы первой и второй плат, при ошибке подключения ко второй плате видеоввода.

OnEvent("GRABBER","2"," UPS_FATAL_ERROR");//ошибка подключения к плате видеоввода 2

{

DoReact("CAM","1","MUX1"); //вывод видеокамеры 1 на 1-ый аналоговый вывод платы

Wait(5);

DoReact("GRABBER","1","MUX1_OFF"); //отключение 1-го аналогового выхода первой платы

DoReact("GRABBER","2","MUX1_OFF"); //отключение 1-го аналогового выхода второй платы

}

Примечание 1. Если аналоговые выходы двух и более плат соединяются параллельно, и видеокамера 1, например, принадлежит первому грабберу, а видеокамера 2 - второму, то при вызове команды «DoReact("CAM","1","MUX1");» необходимо сначала вызвать команду «DoReact("GRABBER","2","MUX1_OFF");» и, соответственно, при вызове команды

«DoReact("CAM","2","MUX1");» необходимо сначала вызвать команду «DoReact("GRABBER","1","MUX1_OFF");». Иначе произойдет наслоение сигналов.

Примечание 2. Описание объекта "CAM" указано ниже (см. раздел «CAM»).

4. Необходимо отключить второй аналоговый выход платы видеоввода при восстановлении питания от сети.

```
OnEvent("GRABBER","1","UPS_ONLINE");        //восстановление питания от сети
{
DoReact("GRABBER","1","MUX2_OFF");          //отключение аналогового выхода 2
}
```

3.7.2 CAM

Объект «CAM» соответствует системному объекту «Камера».

От объекта «CAM» поступают события, представленные в таблице (см. Таб. 3.7-4). Запуск процедур происходит при возникновении соответствующего события. Формат процедуры событий для объекта «Камера»:

```
OnEvent("CAM","_id_","_событие_");
```

Таб. 3.7-4. Описание событий от объекта «CAM»

События	**Описание событий**
"ARM"	Камера поставлена на охрану.
"ATTACH"	Подключение.
"BLINDING"	Камера залеплена.
"DETACH"	Обрыв.
"DISARM"	Камера снята с охраны.
"FILE_REC_ERROR"	Ошибка записи на диск.
"MD_START"	Тревога.
"MD_STOP"	Конец тревоги.
"PRINT"	Печать кадра.
"REC"	Запись на диск.
"REC_STOP"	Остановка записи на диск.
"UNBLINDING"	Камера открыта.

Формат оператора для описания действий с камерой:

```
DoReact("CAM","_id_","_команда_" ["_параметры_"]);
```

Список команд и параметров для объекта «CAM» представлен в таблице (см. Таб. 3.7-5).

Таб. 3.7-5. Список команд и параметров для объекта «CAM»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" - устанавливает (изменяет) параметры камеры.	rec_priority<>	Приоритет записи (от 0 до 3, 0 – обычный, 3 – все ресурсы).
	compression<>	Степень компрессии (0 – компрессия отсутствует, 1- макс. качество, …, 5 – мин. качество).
	sat_u<>	Насыщенность цвета (0 – мин, 10 – макс).
	proc_time<>	Период дозаписи (0 – 30 сек).
	manual<>	Управление настройкой яркости и контрастности (0 – ручное; 1 – автоматическое; 2 – автоматическое, но около значений, выставленных вручную).
	contrast<>	Контрастность (0 – мин, 10 – макс).
	md_size<>	Размер объектов детектора движения (1 -16).
	md_mode<>	Режим записи пауз (1 – включено, 0 выключено).
	audio_type<>	Тип звукового сопровождения.
	pre_rec_time<>	Время предзаписи (0 – 20 сек).
	bright<>	Яркость (0 – мин, 10 – макс).
	audio_id<>	Номер микрофона (пустой параметр, если нет микрофона).
	rec_time<>	Скорость записи (1 – 30 кадров/сек, 0 – не используется).
	alarm_rec<>	Запись тревог (1 – включено, 0 – выключено).
	hot_rec_time<>	Время горячей записи (0 – 30 сек).
	hot_rec_period<>	Период горячей записи (0 – 20 сек).
	mux<>	Номер канала (0 – 1 канал, 15 – 16 канал).
	color<>	Цвет (0 – черно-белый, 1 – цветной).
	activity<>	-
	arch_days<>	Количество дней архива.
	blinding<>	Камера залеплена.
	config_id<>	-
	decoder<>	-
	flags<>	Флаги.

Команда – описание команды	**Параметры**	**Описание параметров**
	fps<>	Скорость записи (0 – не используется, 1 – 30 кадров/сек).
	ifreq<>	Частота опорных кадров в последовательности (1 – каждый кадр опорный, 2 – 100 кадр).
	mask 0, mask1, mask2, mask3, mask4	Маска детектора.
	md_contrast<>	Чувствительность детектора движения (0 – 15).
	motion<>	Оценка движения компрессора (5 - 255).
	name<>	Имя объекта.
	password_crc<>	Пароль на видеоархив.
	priority<>	Приоритет источника постановки на запись (0 – автоматическое, 1 – ручное).
	resolution<>	Разрешение (0 – стандартное CIF, 1 - высокое 2CIF, 2 – максимальное 4CIF).
	type<>	Тип объекта.
	yuv<>	Цветовая схема кодирования видеосигнала (0 – YUV4:2:0, 1 - YUV4:2:2).
"DELETE" - отключает камеру.	-	-
"START_VIDEO" - включает видеопоток для текущей камеры.	slave_id<>	Имя компьютера, к которому подключена камера.
	comress<>	Степень компрессии.
	register_only<>	-
"STOP_VIDEO" - выключает видеопоток для текущей камеры.	slave_id<>	Имя компьютера, к которому подключена камера.
"REQUEST_MASK"	mask<>	Маска.
"MUX1", "MUX2", "MUX3" - вывести изображение камеры на 1, 2, 3 аналоговые выходы.	-	-
"ACTIVATE" - вывести камеру на монитор.	monitor<>	Номер монитора.
"ARM" - поставить камеру на охрану.	-	-
"DISARM" - снять камеру с охраны.	-	-
"REC" - начать запись камеры.	time<>	Время записи в секундах, если равно нулю, - то записывается 1 кадр.
	rollback<>	Если равно 1, то запись производится с откатом.

Команда – описание команды	Параметры	Описание параметров
"REC_STOP" - остановить запись камеры.	-	-
"SET_MASK" - установить маску.	mask<>	Маска.
"ADD_SUBTITLES" - добавить титры.	command<>	Текст накладываемых титров.

Свойства объекта «CAM» показаны в таблице (см. Таб. 3.7-6).

Таб. 3.7-6. Свойства объекта «CAM»

Свойства объекта « CAM »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.
TELEMETRY_ID<>	Идентификатор модуля телеметрии (ID поворотника).
REGION_ID<>	Идентификатор региона.

Примеры использования событий и реакций объекта «Камера»:

1. Необходимо начать запись с первой видеокамеры в цветном режиме, когда она будет поставлена на охрану.

OnEvent("CAM","1","ARM"); //первая видеокамера поставлена на охрану

{

DoReact("CAM","1","SETUP","color<1>"); // запись с первой видеокамеры в цвете

}

2. Необходимо поставить на охрану первую видеокамеру при отключении пятой видеокамеры.

OnEvent("CAM","5","DETACH"); пятая видеокамера отключена

{

DoReact("CAM","1","ARM"); //первая видеокамера поставлена на охрану

}

3. Необходимо использовать половину ресурсов при записи у первой видеокамеры (то есть, если в системе через первую плату видеоввода подключено 4 видеокамеры, то первая будет записывать – со скоростью 6 кадров/сек, а остальные три – по 2 – 2,5 кадра/сек.), если она находится в тревожном состоянии.

OnEvent("CAM","1","MD_START"); //первая видеокамера находится в тревожном состоянии

{

DoReact("CAM","1","SETUP","rec_priority<2>"); // использование половины ресурсов при записи

}

4. Необходимо установить максимальную компрессию синхронно с четвертым микрофоном звуковой платы на первой видеокамере, при записи на диск видео с первой видеокамеры.

OnEvent("CAM","1","REC"); //первая видеокамера ведет запись на диск

{

DoReact("CAM", "1", "SETUP", "compression<5>, audio_type<OLXA_LINE>, audio_id<4>"); //первая видеокамера, максимальная компрессия, синхронно с четвертым микрофоном звуковой платы.

}

5. Необходимо начать запись с первой видеокамеры с минимальным качеством в черно-белом режиме, когда она выйдет из состояния тревоги.

OnEvent("CAM","1","MD_STOP"); // первая видеокамера перестала находиться в тревожном состоянии

{

value = 5;

DoReact("CAM", "1", "SETUP", "compression<" + value + ">,color<0>"); //начать запись первой видеокамеры с минимальным качеством в ч/б режиме.

}

6. Необходимо начать запись с первой видеокамеры в режиме «откат», когда она снята с охраны.

OnEvent("CAM","1","DISARM"); //первая видеокамера снята с охраны

{

DoReact("CAM","1","REC","rollback<1>"); // Начать запись с первой видеокамеры в режиме «откат»

}

7. Установить новые параметры видеоканала при подключении первой видеокамеры.

OnEvent("CAM","1","ATTACH"); //подключена первая видеокамера

{

VIDEO_CANAL_ID = GETOBJECTPARAM("CAM","1","PARENT_ID"); // определяем идентификатор видеоканала, которому принадлежит первая видеокамера

DoReact("GRABBER",VIDEO_CANAL_ID,"SETUP","chan<0>,mode<0>,resolution<1>,format<pal>"); //устанавливаем новые параметры видеоканала.

}

Функция проверки состояния объекта «CAM»:

CheckState("CAM","номер","состояние")

Объект «CAM» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-7).

Таб. 3.7-7. Состояния объекта «CAM»

Состояние объекта «CAM»	**Описание состояния**
"ALARMED"	Камера находится в тревожном состоянии.
"DISARM_DETACHED"	Нет сигнала от камеры.
"DETACHED"	Нет сигнала от камеры.
"ARMED"	Камера поставлена на охрану.
"DISARMED"	Камера снята с охраны.

3.7.3 MONITOR

Объект «MONITOR» соответствует системному объекту «Монитор».

DoReact("MONITOR","_id_","_команда_"[,"_параметры_"]);

Список команд и параметров для объекта «MONITOR» представлен в таблице (см. Таб. 3.7-8).

Таб. 3.7-8. Команды и параметры для объекта «MONITOR»

Команда – описание команды	**Параметры**	**Описание параметров**
"REMOVE" - удаляет камеру с монитора.	cam<>	ID камеры в дереве настроек, которую необходимо удалить с монитора.
"REMOVE_ALL" - удаляет все камеры с монитора.	-	-
"STOP_VIDEO" - останавливает видеопоток камеры.	cam<>	ID камеры в дереве настроек, видеопоток от которой необходимо остановить.
"REPLACE" - удаляет все камеры с монитора и вызывает указанную камеру.	slave_id<>	Имя компьютера, которому принадлежит монитор, в скрипте можно подставить owner.
	cam<>	ID камеры в дереве настроек, которую необходимо вывести на монитор.
	name<>	Название камеры, которое будет отображаться в левом нижнем углу.
	audio_type<>	-
	audio_id<>	-
	arch_id<>	-
	control<>	0 только просмотр архива, 1 – так же возможно и управление (постановка/снятие с охраны, запись).

Команда – описание команды	**Параметры**	**Описание параметров**
"ADD_SHOW" –добавляет камеры на монитор.	cam<>	ID камеры в дереве настроек, которую необходимо вывести на монитор.
	name<>	Имя объекта, которое будет отображаться в левом нижнем углу.
	arch_id<>	-
	control<>	0 только просмотр архива, 1 – так же возможно и управление (постановка/снятие с охраны, запись).
"ACTIVATE_CAM" - делает активной камеру.	cam<>	ID камеры в дереве настроек, которую необходимо сделать активной.
"ARCH_FRAME_TIME" - поиск видеоархива по дате и времени.	cam<>	-
	date<>	-
	time<>	-
"SETUP" – устанавливает параметры монитора.	no_update<>	-
	overlay<>	Включение режима ускорения отображения.
	x<>	Координата левого верхнего угла (0 – 100).
	y<>	Координата левого верхнего угла (0 – 100).
	w<>	Размер по горизонтали (0 – 100).
	h<>	Размер по вертикали (0 – 100).
	max_cams<>	Максимально допустимое число камер на мониторе.
	min_cams<>	Минимально допустимое число камер на мониторе.
	compress<>	-
	panel<>	Показать панель управления (0 – выключена, 1 – включена).
	panel_type<>	-
	s<>	-
	layout<>	-
	gate<>	-
	map_id<>	-
	enable<>	-
	topmost<>	1 - показывать экран поверх всех остальных окон.
	type<>	Тип объекта «Монитор».
	allow_move<>	Разрешить перемещение окна.
	arch_id<>	Идентификатор архива.
	cycle<>	Задержка при автоматическом листании (1 – 20 сек).
	flags<>	Флаги.
	name<>	Имя объекта.
	overlay<>	Включение режима ускорения отображения (0 – нет ускорения, 1 – ускорение «режим Оверлей», 2 – ускорение «режим DirectDraw»).

Команда – описание команды	**Параметры**	**Описание параметров**
	tel_prior<>	Приоритет телеметрии.
"ACTIVATE" - активирование панели управления монитора.	user_id<>	Идентификатор пользователя.
	panel_active<>	-
"DEACTIVATE" - де активирование панели управления монитора.	-	-
"EXPORT_FRAME" - экспорт кадра в JPG-файл.	cam<>	-
	file	-
"KEY_PRESSED" – управление кнопками монитора видеонаблюдения и архива видеозаписей.	number<>	-
	key<>	Возможные значения: "ARCH_EDIT_DATE" – изменить дату поиска по архиву; "ARCH_EDIT_TIME" – изменить время поиска по архиву; "ARCH_EDIT_ENTER" – ввод изменений значений в архиве; "ARCH_EDIT_ESCAPE" – отменить редактирование архива; "ARCH_EDIT_BACK"; "ARCH_EDIT_REPLACE"; "WINDOW_ZOOM_IN" – развернуть окно видеонаблюдения; "WINDOW_ZOOM_OUT" – свернуть окно видеонаблюдения; "ZOOM_IN" – приближение изображения; "ZOOM_OUT" – отдаление изображения; "CYCLE_REW" – вернуться к предыдущему фрагменту; "CYCLE_FF" – перейти к следующему фрагменту; "LEFT" – листание окон видеонаблюдения назад; "RIGHT" – листание окон видеонаблюдения вперед; "UP" – перебор отображаемых окон видеонаблюдения вверх; "DOWN" – перебор отображаемых окон видеонаблюдения вниз; "MODE_VIDEO" – режим видеонаблюдения; "MODE_ARCH" – режим воспроизведения архивных видеозаписей; "MODE_ARCH2"- режим воспроизведения архивных видеозаписей 2; "MASK_SHOW" – нанести маску; "MASK_HIDE" – удалить маску; "ARM" – поставить камеру на охрану; "DISARM" – снять камеру с охраны;

Команда – описание команды	Параметры	Описание параметров
		"REW" – обратная перемотка; "PLAY" – воспроизведение; "PLAY_NONSTOP" – безостановочное воспроизведение; "PLAY_FAST" – ускорить просмотр видеозаписи; "FF" – перемотка вперед; "RECORD" – запись; "RECORD_MIC" – запись с микрофона; "STOP" – остановка; "REC_STOP" – остановка записи; "PAUSE" – пауза; "MIC_ON" – микрофон включен; "MIC_OFF" – микрофон выключен; "PRINT" – вывод кадра на печать. "SELECT_LAYOUT" – управление раскладкой монитора видеонаблюдения

Свойства объекта «MONITOR» показаны в таблице (см. Таб. 3.7-9).

Таб. 3.7-9. Свойства объекта «MONITOR»

Свойства объекта « MONITOR »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Примеры использования событий и реакций объекта «Монитор»:

1. Необходимо при запуске первой макрокоманды проиграть запись с видеокамеры 1 на мониторе 4 с указанными датой и временем.

OnEvent("MACRO","1","RUN");

{

DoReact("MONITOR","4","ARCH_FRAME_TIME","cam<1>,date<"+date+">,time<11:00:00>");

DoReact("MONITOR","4","KEY_PRESSED","key<play>");

}

2. Необходимо при печати кадра с первой видеокамеры, перейти в режим просмотра видеоархива на первой видеокамере монитора 4, и перейти на 10 кадров далее, начиная с фрагмента указанной даты и времени.

OnEvent("CAM", "1", "PRINT");

{

```
DoReact("MONITOR","4","ARCH_FRAME_TIME","cam<1>,date<"+date+">,time <11:00:00>");
for(i=0;i<10;i=i+1)
{
DoReact ("MONITOR","4","KEY_PRESSED","key<ff>");
}}
```

3. Необходимо приблизить видеоизображение на экране монитора, если видеокамера находится в состоянии тревоги, и вернуть в исходное состояние, при ее окончании.

```
OnEvent("CAM","1","MD_START");
{
DoReact("MONITOR","1","KEY_PRESSED","key<zoom_in>");
}
OnEvent("CAM", "1", "MD_STOP");
{
DoReact("MONITOR","1","KEY_PRESSED","key<zoom_out>");
}
```

4. Необходимо вывести на экран монитора раскладку под номером один при срабатывании макрокоманды.

```
OnEvent("MACRO","1","RUN");
{
DoReact("MONITOR","1","KEY_PRESSED","key<SELECT_LAYOUT>,number<1>");
}
```

3.7.4 PLAYER

Объект «PLAYER» соответствует системному объекту «Аудиопроигрыватель».

Формат оператора для описания действий с аудиопроигрывателем:

```
DoReact("PLAYER","_id_","_команда_" ["_параметры_"]);
```

Список команд и параметров для объекта «PLAYER» представлен в таблице (см. Таб. 3.7-10).

Таб. 3.7-10. Команды и параметры для объекта «PLAYER»

Команда – описание команды	**Параметры**	**Описание параметров**
"PLAY_WAV" - проигрывает звуковой файл.	file<>	Звуковой файл с полным путем к нему.
"SETUP" – настройка параметров аудиопроигрывателя.	board<>	Звуковое устройство проигрывателя архива.
	flags<>	Флаги.

Команда – описание команды	Параметры	Описание параметров
	h<>	Высота диалога настройки (0 – 100).
	name<>	Имя объекта.
	voice<>	Звуковое оповещение.
	voice_board<>	Звуковое устройство оповещения.
	w<>	Ширина диалога настройки (0 – 100).
	x<>	Левый верхний угол диалога настройки (0 – 100).
	y<>	Левый верхний угол диалога настройки (0 – 100).

Свойства объекта «PLAYER» показаны в таблице (см. Таб. 3.7-11).

Таб. 3.7-11. Свойства объекта «PLAYER»

Свойства объекта « PLAYER »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Пример использования событий и реакций объекта «Аудиопроигрыватель»:

1. Необходимо проиграть звуковой файл, находящийся по адресу «C:\ Program Files\Intellect\Wav\cam_alarm_1.wav», при включении флага работы аудиопроигрывателя.

OnEvent("PLAYER","1","flags"); // при включении флага работы аудиопроигрывателя

{

DoReact("PLAYER","1","PLAY_WAV","FILE< c:\ program files\intellect\wav\cam_alarm_1.wav >"); // проигрывать звуковой файл

}

3.7.5 OLXA_LINE

Объект «OLXA_LINE» соответствует системному объекту «Микрофон».

От объекта «OLXA_LINE» поступают события, представленные в таблице (см. Таб. 3.7-12). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для микрофона:

OnEvent("OLXA_LINE ", "_id_","_событие_");

Таб. 3.7-12. События от объекта «OLXA_LINE»

Событие	**Описание события**
"ACCU_START"	Включение акустопуска.
"ACCU_STOP"	Выключение акустопуска.
"ARM"	Запись включена.
"DISARM"	Запись выключена.
"INCOMING_NUMBER"	Входящий телефонный номер.
"OUTGOING_NUMBER"	Исходящий телефонный номер.
"REC"	Начало записи.
"REC_STOP"	Конец записи.
"RESET"	Подключение микрофона.

Формат оператора для описания действий с микрофоном:

DoReact("OLXA_LINE ","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «OLXA_LINE» представлен в таблице (см. Таб. 3.7-13).

Таб. 3.7-13. Команды и параметры для объекта «OLXA_LINE»

Команда – описание команды	**Параметры**	**Описание параметров**
"ARM" – включить микрофон на запись.	-	-
"DISARM" – выключить запись с микрофона.	-	-
"SETUP" – настройка параметров микрофона.	type<>	Тип линии.
	accu_start <>	Порог срабатывания детектора звука.
	accu_stop<>	Время удержания сработки детектора.
	amp<>	Усиление.
	aru<>	Автоматическая регулировка усиления.
	aru_dyn<>	Уровень АРУ.
	aru_time<>	Время срабатывания АРУ.
	chan<>	Номер звукового канала микрофона.
	compression<>	Тип компрессии.
	flags<>	Флаги.
	name<>	Имя объекта.
	rec<>	Начало записи.

Свойства объекта «OLXA_LINE» показаны в таблице (см. Таб. 3.7-14).

Таб. 3.7-14. Свойства объекта «OLXA_LINE»

Свойства объекта « OLXA_LINE »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Функция проверки состояния объекта «OLXA_LINE»:

CheckState("OLXA_LINE","номер","состояние")

Объект «OLXA_LINE» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-15).

Таб. 3.7-15. Состояния объекта «OLXA_LINE»

Состояние объекта «OLXA_LINE»	Описание состояния объекта
"BLUE"	Микрофон снят с охраны.
"GREEN"	Нет сигнала от микрофона.
"YELLOW"	Микрофон поставлен на охрану.
"RED"	Начало записи.

Примеры использования событий и реакций объекта «Микрофон»:

1. Необходимо включить первый микрофон на запись при включении акустопуска.

OnEvent("OLXA_LINE","1","accu_start"); //включение акустопуска

{

DoReact("OLXA_LINE","1","ARM"); //включение микрофона на запись

}

2. Необходимо установить минимальную компрессию на микрофоне при выключении записи аудиосигнала.

OnEvent("OLXA_LINE","1","DISARM"); // отключение записи с микрофона

{

DoReact("OLXA_LINE","1","SETUP","compression<5>"); //установлена минимальная компрессия

}

3.7.6 DIALOG

Объект «DIALOG» соответствует системному объекту «Окно запроса оператора».

Формат оператора для описания действий с окном запроса оператора:
DoReact("DIALOG","_id_","_команда_" [,"_параметры_"]);

Список команд и параметров для объекта «DIALOG» представлен в таблице (см. Таб. 3.7-16).

Таб. 3.7-16. Команды и параметры для объекта «DIALOG»

Команда – описание команды	Параметры	Описание параметров
"SETUP" - настройка окна запроса оператора.	x<>	Координата левого верхнего угла (0 - 100).
	y<>	Координата левого верхнего угла (0 - 100).
	allow_move<>	0 – запретить перемещение, 1 – разрешить перемещение.

Команда – описание команды	Параметры	Описание параметров
"RUN" -показать окно запроса оператора.	-	-
"RUN_MODAL" - запуск окна запроса оператора в модальном режиме.	-	-
"CLOSE" - закрывает последнее открытое окно запроса оператора.	-	-
"CLOSE_ALL" - закрывает все открытые окна запроса оператора.	-	-

Примеры использования реакций объекта «Окно запроса оператора»:

1. Необходимо по макрокоманде с номером 1 устанавливать координаты верхнего левого угла окна запроса оператора (поворотной видеокамеры PANASONIC-850) в центре экрана, запрещать его перемещение и выводить его на экран.

```
OnEvent("MACRO","1","RUN")

{

DoReact("DIALOG","PANASONIC-850","SETUP","x<50>,y<50>,allow_move<0>");

DoReact("DIALOG","PANASONIC-850","RUN");

}
```

2. Необходимо закрывать окно запроса оператора по макрокоманде с номером 2.

```
OnEvent("MACRO","2","RUN")

{

DoReact("DIALOG","PANASONIC-850","CLOSE");

}
```

3.7.7 MMS

Объект «MMS» соответствует системному объекту «Сервис почтовых сообщений».

От объекта «MMS» поступают события, представленные в таблице (см.Таб. 3.7-17). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для сервиса почтовых сообщений:

```
OnEvent("MMS","_id_","_событие_");
```

Таб. 3.7-17. События от объекта «MMS»

Событие	Описание события
"SET_CONNECTIONS"	Список доступных подключений.

Формат оператора для описания действий с сервисом почтовых сообщений:

DoReact("MMS","_id_","_команда_" [,"_параметры_"]);

Список команд и параметров для объекта «MMS» представлен в таблице (см. Таб. 3.7-18).

Таб. 3.7-18. Команды и параметры для объекта «MMS»

Команда – описание команды	Параметры	Описание параметров
"SETUP" - настройки для сервиса почтовых сообщений.	smtp<>	Адрес SMTP сервера.
	connection<>	Тип подключения.
	smtp_username<>	Имя пользователя.
	smtp_password<>	Пароль.
	port<>	Номер порта.
	flags<>	Флаги.
	name <>	Название объекта.
"GET_CONNECTIONS" - получить список доступных подключений.	-	-

Свойства объекта «MMS» показаны в таблице (см. Таб. 3.7-19).

Таб. 3.7-19. Свойства объекта «MMS»

Свойства объекта « MMS »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Пример использования реакций объекта «Сервис почтовых сообщений»:

1. Необходимо установить номер порта почтовой службы – 25 при выполнении макрокоманды 1.

OnEvent("MACRO","1","RUN");

{

DoReact("MMS", "1", "SETUP", "port<25>");

}

3.7.8 MAIL_MESSAGE

Объект «MAIL_MESSAGE» соответствует системному объекту «Почтовое сообщение».

От объекта «MAIL_MESSAGE» поступают события, представленные в таблице (см. Таб. 3.7-20). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для почтового сообщения:

OnEvent("MAIL_MESSAGE","_id_","_событие_");

Таб. 3.7-20. События от объекта «MAIL_MESSAGE»

Событие	**Описание события**
“SEND_ERROR”	Ошибка отправки сообщения.
“SENT”	Сообщение отправлено.

Формат оператора для описания действий с почтовым сообщением:

DoReact("MAIL_MESSAGE","_id_","_команда_" [,"_параметры_"]);

Список команд и параметров для объекта «MAIL_MESSAGE» представлен в таблице (см. Таб. 3.7-21).

Таб. 3.7-21. Команды и параметры для объекта «MAIL_MESSAGE»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" - настройки для почтового сообщения.	from<>	Адрес отправителя.
	to<>	Адрес получателя.
	cc<>	Копии.
	subject<>	Тема сообщения.
	body<>	Тело сообщения.
	attachments<>	Приложения.
	flags<>	Флаги.
	name<>	Имя объекта.
	pack<>	Способ упаковки приложений.
"SEND" – отправка почтового сообщения.	-	-

Свойства объекта «MAIL_MESSAGE» показаны в таблице (см. Таб. 3.7-22).

Таб. 3.7-22. Свойства объекта «MAIL_MESSAGE»

Свойства объекта « MAIL_MESSAGE »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Пример использования реакций объекта «Почтовое сообщение».

1. Необходимо отправить сообщение при срабатывании датчика движения вместе с картинкой от видеокамеры при переходе видеокамеры в состояние тревоги.

OnEvent("CAM",N,"MD_START"); //видеокамера в состоянии тревоги

{

filename = "c:\" + N + "_msg_" + i + ".jpg";

DoReact("MONITOR","1","EXPORT_FRAME","cam<" + N + ">,file<" + filename+ ">");

DoReact("MAIL_MESSAGE","1","SETUP","body<сработала камера"+ n + ">,subject<тревога по камере>,from<sergey.kozlov@itv.ru>, to<sergey.kozlov@itv.ru>,attachments<" + filename +">");

DoReact("MAIL_MESSAGE","1","SEND");

}

3.7.9 VMS

Объект «VMS» соответствует системному объекту «Сервис голосовых сообщений».

Формат оператора для описания действий с сервисом голосовых сообщений:

DoReact("VMS","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «VMS» представлен в таблице (см. Таб. 3.7-23).

Таб. 3.7-23. Команды и параметры для объекта «VMS»

Команда – описание команды	**Параметры**	**Описание параметров**
"SEND" – послать сообщение.	modem<>	Название устройства.
	pulse<>	Тип набора (0 – тоновый, 1 – импульсный).
	name<>	Имя объекта.
	redial_attempts<>	Количество попыток дозвона.
	redial_delay<>	Пауза между попытками дозвона.
	waitfordialtone<>	Ожидание сигнала линии (0 - нет, 1 – да).
	flags<>	Флаги.

Свойства объекта «VMS» показаны в таблице (см. Таб. 3.7-24).

Таб. 3.7-24. Свойства объекта «VMS»

Свойства объекта « VMS »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Пример использования реакций объекта «Сервис голосовых сообщений»:

1. Необходимо при выполнении макрокоманды 1 послать сообщение, если модем подключен к порту COM2, тип набора – импульсный, не дожидаться тонального сигнала.

```
OnEvent("MACRO","1","RUN")

{

DoReact("VMS","1","SEND","modem<2>,pulse<1>,waitfordialtone<0>");

}
```

3.7.10 GRELE

Объект «GRELE» соответствует системному объекту «Реле».

От объекта «GRELE» поступают события, представленные в таблице (см. Таб. 3.7-25). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для реле:

```
OnEvent("GRELE", "_id_","_событие_");
```

Таб. 3.7-25. События от объекта «GRELE»

Событие	**Описания события**
"OFF"	Реле выключено.
"ON"	Реле включено.
"SIGNAL_LOST"	Потеря связи.

Формат оператора для описания действий с реле:

Формат: DoReact("GRELE","_id_","_команда_");

Список команд и параметров для объекта «GRELE» представлен в таблице (см. Таб. 3.7-26).

Таб. 3.7-26. Команды и параметры для объекта «GRELE»

Команда – описание команды	**Параметры**	**Описание параметров**
"ON" - включить реле.	-	-

Команда – описание команды	Параметры	Описание параметров
"OFF" - выключить реле.	-	-
"SETUP" – настройки для реле.	chan <>	Номер выхода (0 – 15).
	flags<>	Флаги.
	name<>	Имя объекта.

Свойства объекта «GRELE» показаны в таблице (см. Таб. 3.7-27).

Таб. 3.7-27. Свойства объекта «GRELE»

Свойства объекта « GRELE »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.
REGION_ID<>	Идентификатор региона.

Функция проверки состояния объекта «GRELE»:

CheckState("GRELE","номер", "состояние")

Объект «GRELE» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-28).

Таб. 3.7-28. Состояния объекта «GRELE»

Состояние объекта «GRELE»	Описание состояния объекта
"ON"	Реле включено.
"OFF"	Реле выключено.
"DETACHED_ON"	Потеря связи.
"DETACHED_OFF"	Потеря связи.

Примеры использования событий и реакции объекта «Реле»:

1. Необходимо при потере связи с реле 1 включить реле 2.

```
OnEvent("GRELE","1","SIGNAL_LOST");
{
DoReact("GRELE", "2", "ON");
}
```

3.7.11 GRAY

Объект «GRAY» соответствует системному объекту «Луч».

От объекта «GRAY» поступают события, представленные в таблице (см. Таб. 3.7-29). Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для луча:

OnEvent("GRAY","_id_","_событие_");

Таб. 3.7-29. События от объекта «GRAY»

Событие	**Описание события**
"ALARM"	Тревога.
"ARM"	Луч поставлен на охрану.
"CONFIRM"	Тревога принята.
"DISARM"	Луч снят с охраны.
"NOT_VALID_STATE"	Зона не готова.
"OFF"	Луч разомкнут.
"ON"	Луч замкнут.
"SIGNAL_LOST"	Потеря связи с лучом.

Формат оператора для описания действий с лучом:

DoReact("GRAY","_id_","_команда_");

Список команд и параметров для объекта «GRAY» представлен в таблице (см. Таб. 3.7-30).

Таб. 3.7-30. Команды и параметры для объекта «GRAY»

Команда – описание команды	**Параметры**	**Описание параметров**
"ARM" – поставить на охрану луч.	-	-
"DISARM" – снять с охраны луч.	-	-
“CONFIRM” – принять тревогу.	-	-
“SETUP” – настройки для луча.	chan<>	Номер входа (0 – 15).
	flags<>	Флаг и.
	name<>	Имя объекта.
	type<>	Тип объекта луч (0 – на замыкание, 1 – на размыкание).

Свойства объекта «GRAY» показаны в таблице (см. Таб. 3.7-31).

Таб. 3.7-31. Свойства объекта «GRAY»

Свойства объекта « GRAY »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.
REGION_ID<>	Идентификатор региона.

Функция проверки состояния объекта «GRAY»:

CheckState ("GRAY","номер","состояние")

Объект «GRAY» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-32).

Таб. 3.7-32. Состояния объекта «GRAY»

Состояние объекта «GRAY»	Описание состояния объекта
"ARMED"	Луч поставлен на охрану.
"DISARME"”	Луч снят с охраны.
"ALARMED"	Тревога.
"CONFIRMED"	Тревога принята.
"DISARMED_ALARM"	Неготовность.
"DETACHED_ARMED"	Потеря связи.
"DETACHED_DISARM"	Потеря связи.

Примеры использования событий и реакций объекта «Луч»:

1. Необходимо перевести второй луч на второй вход, если потеряна связь с первым лучом.

OnEvent("GRAY","1"," SIGNAL_LOST"); //потеряна связь с первым лучом

{

DoReact("GRAY","2","SETUP","chan<2>"); //луч на втором входе

}

2. Необходимо разомкнуть второй луч и поставить на запись с откатом первую видеокамеру, в случае, когда первый луч замкнут.

OnEvent("GRAY","1"," ON"); //первый луч замкнут

{

DoReact("GRAY","2","SETUP","type<1>"); //разомкнуть второй луч

DoReact("CAM","1","REC","rollback<1>");//производится запись с откатом с первой видеокамеры

}

3.7.12 VNS

Объект «VNS» соответствует системному объекту «Сервис голосового оповещения».

Формат оператора для описания действий с сервисом голосового оповещения:

DoReact("VNS","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «VNS» представлен в таблице (см. Таб. 3.7-33).

Таб. 3.7-33. Команды и параметры для объекта «VNS»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – настройка сервиса голосового оповещения.	card<>	Имя звукового устройства. *Примечание. Имя карты должно строго соответствовать тому названию, что указано в настройках звуковой карты «Сервиса голосового оповещения» системы «Интеллект».*
	level<>	Уровень сигнала. Значение параметра варьируется от 0 до 15. По умолчанию оно равно 8, то есть среднему.
	channel<>	Набор звуковых каналов. Возможные значения параметра: 0 – нет звукового канала; 1 – левый канал воспроизведения; 2 – правый канал воспроизведения; 3 – левый и правый канал воспроизведения (оба канала).
	flags<>	Флаги.
	ip<>	IP-адрес сетевого устройства.
	name<>	Имя объекта.
	pass<>	Пароль.
	user<>	Имя пользователя.
"PLAY" – проигрывание звукового файла.	file<>	Полный путь и имя звукового файла. *Примечание. Если указано только имя файла, то путь к нему по умолчанию будет взят с реестра, с раздела «HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect», в значении параметра «InstallPath». Также в данном параметре есть возможность проигрывания нескольких музыкальных файлов с помощью операции «+».*

Свойства объекта «VNS» показаны в таблице (см. Таб. 3.7-34).

Таб. 3.7-34. Свойства объекта «VNS»

Свойства объекта « VNS »	Описание свойства объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Примеры использования событий и реакций объекта «Сервис голосового оповещения»:

1. Необходимо проигрывать звуковой файл при начале записи видеокамеры.

```
OnEvent("CAM",N,"REC")

{


DoReact("VNS","1","PLAY","file<C:\Program Files\ Intellect\Wav\cam_alarm_"+N+".wav>");

}
```

2. Необходимо, чтобы при наступлении, заранее заданной временной зоны, менялось значение регулятора громкости на меньшее, а затем по её окончании, ставилось значение равному среднему.

```
OnEvent("TIME_ZONE","1","ACTIVATE")

{

DoReact("VNS","1","SETUP","level<2>");

}

OnEvent("TIME_ZONE","1","DEACTIVATE")

{

DoReact("VNS","1","SETUP","level<8>");

}
```

Примечание. Описание объекта "TIME_ZONE" указано ниже (см. раздел «TIME_ZONE»).

3.7.13 SMS

Объект «SMS» соответствует системному объекту «Сервис коротких сообщений».

Формат оператора для описания действий с сервисом коротких сообщений:

```
DoReact("SMS","_id_","_команда_" ["_параметры_"]);
```

Список команд и параметров для объекта «SMS» представлен в таблице (см. Таб. 3.7-35).

Таб. 3.7-35. Команды и параметры для объекта «SMS»

Команда – описание команды	Параметры	Описание параметров
"SETUP" – настройка сервиса коротких сообщений.	device<>	SMS устройство.
	flags<>	Флаги.
	message<>	Текст сообщения.
	name<>	Имя объекта.
	phone<>	Номер телефона.

Свойства объекта «SMS» показаны в таблице (см. Таб. 3.7-36).

Таб. 3.7-36. Свойства объекта «SMS»

Свойства объекта « SMS »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Примеры использования событий и реакций объекта «Сервис коротких сообщений»:

1. Необходимо послать короткое сообщение на номер «89179190909» при тревоге на первой видеокамере.

OnEvent("CAM","1","MD_START");

{

DoReact("SMS","1","SETUP","phone<+79179190909>,message<камера 1, тревога>");

}

2. Необходимо установить устройство для передачи коротких сообщений и послать сообщение по номеру «89179190909», при тревоге на первом луче.

OnEvent("GRAY","1","CONFIRM"); //принять тревогу от луча 1

{

DoReact("SMS","1","SETUP","device<>",); //установить устройство для передачи коротких сообщений

DoReact("SMS","1","SETUP","phone<+79179190909>,message<луч 1, тревога>"); //послать сообщение о тревоге на луче 1 по номеру телефона

}

3.7.14 TELEMETRY

Объект «TELEMETRY» соответствует системному объекту «Поворотное устройство».

Формат оператора для описания действий с поворотными устройствами:

DoReact("TELEMETRY ","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «TELEMETRY» представлен в таблице (см. Таб. 3.7-37).

Таб. 3.7-37. Команды и параметры для объекта «TELEMETRY»

Команда – описание команды	**Параметры**	**Описание параметров**
"AUTOFOCUS_ON" – включение функции автонаведения.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"AUTOPAN_END_P" – задание конечной точки автоповорота.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"AUTOPAN_START" – начать автоповорот.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"AUTOPAN_START_P" – задание стартовой точки автоповорота.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"AUTOPAN_STOP" – окончить автоповорот.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"CLEAR_PRESET" – очистить выбранный пресет.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
	preset<>	Пресет.
"D2OFF" – отключение дополнительных динамических настроек для поворотных видеокамер Panasonic, предназначенных для улучшения качества аналогового видеосигнала.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"D2ON" – включение дополнительных динамических настроек для поворотных видеокамер Panasonic, предназначенных для улучшения качества аналогового видеосигнала.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"DOWN" – повернуть объектив видеокамеры вниз.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"FOCUS_IN" – увеличить изображение.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"FOCUS_OUT" – уменьшить изображение.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"FOCUS_STOP" – остановить увеличение/уменьшение изображения.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"GO_PRESET" – повернуть видеокамеру в положение, заданное на пресете.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
	preset<>	Пресет.

Команда – описание команды	**Параметры**	**Описание параметров**
"HOME" – повернуть видеокамеру в исходную (домашнюю) позицию.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"IRIS_CLOSE" – закрыть диафрагму.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"IRIS_OPEN" – открыть диафрагму.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"IRIS_STOP" – остановить диафрагму.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"LEFT" – повернуть объектив видеокамеры влево.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"LEFT_DOWN" – повернуть объектив видеокамеры влево и вниз.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"LEFT_UP" – повернуть объектив видеокамеры влево и вверх.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"PATROL_LEARN" – начать процедуру программирования патрулирования, выполняемую путем записи поведения видеокамеры.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"PATROL_PLAY" – начать патрулирование.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"PATROL_STOP" – закончить патрулирование.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"RIGHT" – повернуть объектив видеокамеры вправо.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"RIGHT_DOWN" – повернуть объектив видеокамеры вправо и вниз.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"RIGHT_UP" – повернуть объектив видеокамеры вправо и вверх.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"SET_PRESET" – записать текущее положение видеокамеры в выбранный пресет.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
	preset<>	Пресет.
"STOP" – завершить поворот объектива видеокамеры.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"UP" – повернуть объектив видеокамеры вверх.	tel_prior<>	Приоритет (1 - низкий, 2 – средний, 3 – высокий).
"SETUP" – настройка поворотного устройства.	address<>	Адрес устройства.
	cam<>	Идентификатор камеры для управления.
	flags<>	Флаг работы объекта (0 – включен, 1 - отключен).

Команда – описание команды	Параметры	Описание параметров
	name<>	Имя объекта поворотного устройства.
	speed<>	Скорость.

Свойства объекта «TELEMETRY» показаны в таблице (см. Таб. 3.7-38).

Таб. 3.7-38. Свойства объекта «TELEMETRY»

Свойства объекта «TELEMETRY»	Описание свойств объекта
ID<>	Идентификатор объекта поворотного устройства.
PARENT_ID<>	Идентификатор родительского объекта.

Примеры использования реакций объекта «TELEMETRY»:

1. Необходимо установить автофокусирование, когда видеокамеру ставят на охрану.

```
OnEvent("CAM","1","ARM");

{

DoReact("TELEMETRY","1", "AUTOFOCUS_ON");

}
```

2. Необходимо повернуть видеокамеру в положение заданное в первом пресете, при включении реле.

```
OnEvent("GRELE","1","ON");

{

telemetry_id= GetObjectParam("CAM","1","parent_id");

DoReact("TELEMETRY","telemetry_id","SETUP","GO_preset<1>");

}
```

3.7.15 MACRO

Объект «MACRO» соответствует системному объекту «Макрокоманда».

От объекта «MACRO» поступают события, представленные в таблице (см. Таб. 3.7-39). Запуск процедур происходит при возникновении соответствующего события. Формат процедуры событий для объекта «Макрокоманда»:

```
OnEvent("MACRO","_id_","_событие_");
```

Таб. 3.7-39. События от объекта «MACRO»

События	Описание событий
"RUN"	Выполнено действие.

Формат оператора для описания действий с макрокомандами:

DoReact("MACRO","_id_","_команда_" [,"_параметры_"]);

Список команд и параметров для объекта «MACRO» представлен в таблице (см. Таб. 3.7-40).

Таб. 3.7-40. Команды и параметры для объекта «MACRO»

Команда – описание команды	Параметры	Описание параметров
"RUN" – выполнить действие	-	-
"SETUP" – установить параметры для макрокоманды	name<>	Имя объекта.
	flags<>	Флаги.
	state<>	Состояние объекта.
	hidden<>	Флажок «Скрытый».
	local<>	Флажок «Локальный».

Свойства объекта «MACRO» показаны в таблице (см. Таб. 3.7-41).

Таб. 3.7-41. Свойства объекта «MACRO»

Свойства объекта « MACRO »	Описание свойств объекта
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Функция проверки состояния объекта «MACRO»:

CheckState ("MACRO","номер","состояние")

Объект «MACRO» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-42).

Таб. 3.7-42. Состояния объекта «MACRO»

Состояние объекта «MACRO»	Описание состояния объекта
"NORM"	Норма.

Примеры использования событий и реакций объекта «MACRO»:

1. Необходимо записать текущее положение видеокамеры в 1 пресет, при выполнении макрокоманды 1.

OnEvent("MACRO","1","RUN");

{

DoReact("TELEMETRY","1","SET_PRESET","TEL_PRIOR<1>");

}

2. Необходимо выполнить макрокоманду 2, если камера поставлена на охрану.

OnEvent("CAM","1","ARM");

{

DoReact("MACRO", "2", "RUN");

}

3.7.16 TIME_ZONE

Объект «TIME_ZONE» соответствует системному объекту «Временная зона».

От объекта «TIME_ZONE» поступают события, представленные в таблице (см. Таб. 3.7-43). Запуск процедур происходит при возникновении соответствующего события. Формат процедуры событий для объекта «Временная зона»:

OnEvent("TIME_ZONE","_id_","_событие_");

Таб. 3.7-43. События от объекта «TIME_ZONE»

События	**Описание событий**
"ACTIVATE"	Начало.
"DEACTIVATE"	Конец.

Формат оператора для описания действий с временной зоной:

DoReact("TIME_ZONE","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «TIME_ZONE» представлен в таблице (см. Таб. 3.7-44).

Таб. 3.7-44. Команды и параметры для объекта «TIME_ZONE»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – установить параметры для временной зоны	name<>	Имя объекта.
	flags<>	Флаги.

Свойства объекта «TIME_ZONE» показаны в таблице (см. Таб. 3.7-45).

Таб. 3.7-45. Свойства объекта «TIME_ZONE»

Свойства объекта « TIME_ZONE »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Функция проверки состояния объекта «TIME_ZONE»:

CheckState ("TIME_ZONE","номер","состояние")

Объект «TIME_ZONE» может находиться в состояниях, описанных в таблице (см. Таб. 3.7-46).

Таб. 3.7-46. Состояния объекта «TIME_ZONE»

Состояние объекта «TIME_ZONE»	**Описание состояния объекта**
"ACTIVATE"	Активный.
"INACTIVE"	Неактивный.

Примеры использования событий и реакций объекта «TIME_ZONE»:

1. При активировании первой временной зоны вывести на монитор видеоизображение с камеры №1.

OnEvent("TIME_ZONE","1","ACTIVATE");

{

DoReact ("CAM", "1", "ACTIVATE", "MONITOR<1>");

}

3.7.17 SSS_WATCHDOG

Объект «SSS_WATCHDOG» соответствует системному объекту «Служба перезагрузки системы».

От объекта «SSS_WATCHDOG» поступают события, представленные в таблице (см. Таб. 3.7-47). Запуск процедур происходит при возникновении соответствующего события. Формат процедуры событий для объекта «Служба перезагрузки системы»:

OnEvent("SSS_WATCHDOG","_id_","_событие_");

Таб. 3.7-47. События от объекта «SSS_WATCHDOG»

События	**Описание событий**
"RESTART_EXCEEDED"	Превышено количество перезагрузок модуля.
"RESTART_PROCESS"	Перезагрузка модуля.

Формат оператора для описания действий со службой перезагрузки системы:

DoReact("SSS_WATCHDOG","_id_","_команда_" [,"_параметры_"]);

Список команд и параметров для объекта «SSS_WATCHDOG» представлен в таблице (см. Таб. 3.7-48).

Таб. 3.7-48. Команды и параметры для объекта «SSS_WATCHDOG»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – установить параметры для службы перезагрузки системы.	name<>	Имя объекта.
	flags<>	Флаги.
	restart_period<>	Период рестарта.
	restart_times<>	Максимальное число перезагрузок за заданный промежуток времени.
	timeout<>	Время отклика.
	usb_wd_control<>	Подключение ITV USB Watchdog.

Свойства объекта «SSS_WATCHDOG» показаны в таблице (см. Таб. 3.7-49).

Таб. 3.7-49. Свойства объекта «SSS_WATCHDOG»

Свойства объекта « SSS_WATCHDOG »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.

Примеры использования событий и реакций объекта «SSS_WATCHDOG»:

1. При перезагрузке модуля активировать третью камеру на монитор №5.

OnEvent("SSS_WATCHDOG","1"," RESTART_PROCESS");

{

DoReact("MONITOR", "5", " ACTIVATE_CAM", "CAM<3>")

}

3.7.18 SLAVE

Объект «SLAVE» соответствует системному объекту «Компьютер».

От объекта «SLAVE» поступают события, представленные в таблице (см. Таб. 3.7-50). Запуск процедур происходит при возникновении соответствующего события. Формат процедуры событий для объекта «Компьютер»:

OnEvent("SLAVE","_id_","_событие_");

Таб. 3.7-50. События от объекта «SLAVE»

События	**Описание событий**
CONNECTED	Подключение.
DISCONNECTED	Отключение.
KEY_IGNORED_HW	Ключ отвергнут (несоответствие кодов плат).
KEY_IGNORED_SW	Ключ отвергнут (превышено ограничение).
KEY_UPDATED	Ключ обновлен.
PROTOCOL_RCVD	Протокол получен.
REBUILD_IN_START	Начало переиндексации архива.
REBUILD_IN_STOP	Окончание переиндексации архива.
REGISTER_ATTEMPT	Попытка несанкционированного доступа.
REGISTER_ERROR	Превышен лимит попыток доступа.
REGISTER_USER	Регистрация пользователя.
DISC_EXIST	Диск для записи архива присутствует.
NO_DISC	Диск для записи архива отсутствует.
KEY_IGNORED_FR	Ключ отвергнут.
SHUTDOWN	Завершение работы.

Формат оператора для описания действий с объектом компьютер:

DoReact("SLAVE","_id_","_команда_" ["_параметры_"]);

Список команд и параметров для объекта «SLAVE» представлен в таблице (см. Таб. 3.7-51).

Таб. 3.7-51. Команды и параметры для объекта «SLAVE»

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – установить параметры для компьютера.	display_id<>	Идентификатор экрана.
	drives<>	Диски для записи видеоархива.
	drives_a<>	Диски для записи аудиоинформации.

	flags<>	Флаги.
	arch_days<>	Размер архива событий.
	connection<>	Соединение.
	disable_protocol<>	Отключить протоколирование.
	ip_address<>	IP адрес устройства.
	is_backup<>	Архивация.
	is_load<>	Загружен.
	local_protocol<>	Локальный протокол.
	modem<>	Модемное соединение.
	name<>	Имя объекта.
	password<>	Пароль.
	sync_time<>	Синхронизировать время.
	username<>	Имя пользователя.
"BACKUP" – сделать резервную копию БД.	-	-
"CONNECT_ONE" – подключиться к компьютеру.	-	-
"CONNECT_OTHER" – подключиться к ядрам.	-	-
"DISCONNECT_ONE" – отключиться от компьютера.	-	-
"SYNC_PROTOCOL" – получить протоколы.	-	-
"SYNC_TIME" – синхронизовать время.	-	-
"CREATE_PROCESS" – запустить процесс.	-	-
"SEND_MY_CONFIG" – разослать конфигурацию.	-	-

Свойства объекта «SLAVE» показаны в таблице (см. Таб. 3.7-52).

Таб. 3.7-52. Свойства объекта «SLAVE»

Свойства объекта « SLAVE »	**Описание свойств объекта**
ID<>	Идентификатор объекта.
PARENT_ID<>	Идентификатор родительского объекта.
USER_ID<>	Идентификатор пользователя.

Примеры использования событий и реакций объекта «SLAVE»:

1. При отсутствии диска для записи архива, остановить запись с камеры №2.

```
OnEvent("SLAVE","1"," NO_DISC");
{
        DoReact("CAM","2"," REC_STOP")
}
```

4 Заключение

Пожелания и замечания по данному Руководству следует направлять в Отдел технического документирования компании «Ай-Ти-Ви групп» (documentation@itv.ru).