Ай-Ти-Ви групп

Программный комплекс «Интеллект»
Руководство по программированию

Версия 1.5

Москва
2008

Содержание

1 Введение

Программирование в системе Интеллект предназначено для более гибкого управления объектами, а также управления взаимодействия между объектами системы Интеллект.

2 Инструментарий программирования

2.1 Отладочное окно

В системе "Интеллект" существует возможность в реальном времени просматривать все события и реакции, происходящие в системе. Для этого служить отладочное окно, которое можно включить\выключить через реестр Windows или с помощью утилиты tweaki.exe

Включение Отладочного окна

1. Через реестр Windows

По пути в реестре **HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect**

Изменить строковый параметр Debug в значение "1", "2" или "3".

2. Через утилиту Tweaki.exe (по пути :\\Intellect\tools)

В разделе Intellect, значение Debug Mode выставить равным "1" или "2"

Выключение Отладочного окна

1. Через реестр Windows

По пути в реестре **HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect**

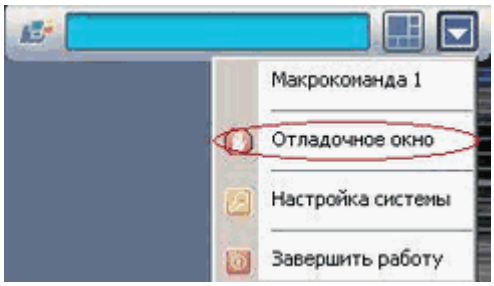
Изменить строковый параметр Debug в значение "None"

2. Через утилиту Tweaki.exe (по пути :\\Intellect\tools)

В разделе Intellect, значение Debug Mode выставить равным "0"

Работа с Отладочным окном

При включенном Отладочным окне в реестре, при следующем запуске системы Интеллект в главном меню системы появится дополнительная команда, которая включает и выключает Отладочное окно при работающей системе:



Если включить эту команду, то появится само отладочное окно:

Свойства Отладочного окна

1. Отладочное окно может менять свои размеры с помощью мыши.

2. Отладочное окно всегда находится поверх других окон.

3. Чтобы прочитать и скопировать строчку из отладочного окна, активируйте эту строчку и нажмите правую клавишу мыши.

Обязательно закройте это добавочное окно, если его не используете.

2.2 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

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

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

2.3 Рекомендуемый порядок написания программ

1. Постановка общей задачи.

2. Разбитие задачи на подзадачи.

3. Написание подзадач и их отладка.

4. Поиск и исправление ошибок

Постановка общей задачи

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

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.

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.

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 (стока1,строка2). Функция "boolstrequal" возвращает значение, отличное от нуля, если строки равны (см. пункт "Описание функций").

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

3.2 Описание процедур

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

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
}
```

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
}
```

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 сработке детектора движения на камере 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 параметрыми 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

}

}
```

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:

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

Пример использования

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

{

DoReact("GRELE","1","ON");

//при замыкании луча 1 замкнуть реле1

}
```

DoCommand(командная строка) – запуск командной строчки

Пример использования

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

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 сек.

Пример использования

```
OnEvent("GRAY","1","ON")
{
[
DoReact("GRELE","1","ON");
Wait(5);
DoReact("GRELE","1","OFF");
// При замыкании Луча 1 Реле 1 будет замыкаться на 5 сек.
]
}
```

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 использования

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

{

if(CheckState("CAM","2","ALARMED"))

{

DoReact("GRELE","1","ON");

//при замыкание луча 1 проверяется состояние камеры 2 и если состояние «Тревога», то замкнуть реле1

}

}

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;}

else {y=3;}

// если "x" больше чем 10 то y=2 иначе y=3

}

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

Примеры использования

```
OnEvent ("MACRO","1","RUN")
{
// постановка 3х зон болида на охрану при запуске макрокоманды 1
for(x=1;x<4;x=str(x+1))
{
DoReact ("BOLID_ZONE",x,"ARM");
}
}
OnEvent ("GRAY","1","ON")
{
// При замыкании луча 1 реле 1 будет замыкаться и размыкаться с с интервалом в 1 сек и это будет происходить 10 раз
[
for(i=0;i<10;i=str(i+1))
{
  DoReact("GRELE","1","ON");
  Wait(1);
  DoReact("GRELE","1","OFF");
  Wait(1);
}
]
}
```

3.4 Операции и выражения

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:
	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;

Оператор	Общее описание, пример использования

	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 логическое И. Пример использования: OnEvent ("MACRO","1","RUN")

Оператор	Общее описание, пример использования

	{
	a=1;
	b=2;
	z=3;
	if((a<b)&&(b<z)) {y=1; //если лож , 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

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 (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<21-09-04>
cos[1]	Тригонометрическая функция расчета косинуса угла. Формат: y=cos(x); где: y-значение функции, x-аргумент (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 (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:

Функции *(В квадратных скобках указано количество исполняемых параметров)*	**Общее описание, пример использования**
	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>
sinh[1]	Функция sinh возвращает гиперболический синус значения аргумента. Формат: y=sinh(x); где: y-значение функции, x-аргумент 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 Пример:

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

Функции *(В квадратных скобках указано количество исполняемых параметров)*	**Общее описание, пример использования**
rad[1]	Тригонометрическая функция расчета угла. Формат: y=rad(x); где: y-значение функции в радианах, x-аргумент 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>
ПРЕОБРАЗОВАНИТЕЛИ	
floor[1]	Функция преобразования до целого числа в меньшую сторону. Формат: x= floor(y); где: x-значение функции, y- дробное или целое число. Пример: 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);

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

	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> Event : CORE VAR_CHANGED value<12>,name<x>,time<15:34:44>,date<17-03-08>

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

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

Функции *(В квадратных скобках указано количество исполняемых параметров)*	**Общее описание, пример использования**
	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=time2 long(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<22-09-04>
	
	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 Формат: x=scalar(y); где: x- числовое значение (в сутках), y- дата. Формат записи: <ДД.ММ.ГГГГ> Пример: x=scalar(19.01.2004) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<731599>,name<x>,time<12:33:29>,date<23-09-04>

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

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

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

	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>
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>

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

Функции *(В квадратных скобках указано количество исполняемых параметров)*	**Общее описание, пример использования**
	* "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<23-09-04>
	
	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.

straright[2]	Выравнивание вправо Формат: x=straright(y,z); где: x-выровненная строка, y-строка, z-значение выравнивания.

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

	Пример 1:
	
	y=str(52657447);
	
	x=straright(y,5);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED value<52657447>,name<y>,time<14:32:27>,date<13-10-04>
	
	Event : CORE VAR_CHANGED value<57447>,name<x>,time<14:32:27>,date<13-10-04>
	
	Пример 2:
	
	x=straright("end",6);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED value< end>,name<x>,time<16:29:02>,date<13-10-04>
	
	Примечание. В случае, если число z больше, чем количество символов в строке, функция дополняет исходную строку пробелами слева 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>

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

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

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

	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)
	
	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>

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

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:

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

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

Задача. Выводить активную камеру на аналоговый монитор

Реализация:

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

Пример 2

Задача. 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

Задача. 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 вечного цикла и выхода из него (переменная run – условие выхода из цикла). Старт цикла по макрокоманде1, остановка по макрокоманде 2. Пока цикл работает, переменная i меняет значение от 1 до 10.

Реализация:

```
OnEvent("MACRO","1","RUN")
{
run=1;
for(i=1;run;i=str(i+1))
{
if (strequal(i,"10")) {i =0;}
}
}

OnEvent("MACRO","2","RUN")
{
run=0;
}
```

Пример 5

По макросу 1 оперативный архив заберет архив по камере 1 за 26.02.06. Если камер больше - добавляются соответсвующие строчки

Реализация:

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



{

DoReact("ARCH","2","START","datetime_from<26-02-06 00:00:00>,cam<1>,datetime_to<26-02-06 23:59:59>");

}
```

Пример 6

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 7

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 8

Задача. 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 9

Есть 2 экрана, первый отображает виртуальный монитор с камерами, второй отображает объект «Карта» с датчиками ОПС «Болид»

Задача. При сработке тревоги по камере – показывается Экран 1, при сработке тревоги от датчика – показывается Экран 2, но только на компьютере CLIENT.

Реализация:

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

{

DoReact("DISPLAY","2","DEACTIVATE","macro_slave_id<CLIENT>");

DoReact("DISPLAY","1","ACTIVATE","macro_slave_id< CLIENT >");

}


OnEvent("BOLID_ZONE",M,"ALARM")

{

DoReact("DISPLAY","1","DEACTIVATE","macro_slave_id< CLIENT >");

DoReact("DISPLAY","2","ACTIVATE","macro_slave_id< CLIENT >");

}
```

Пример 10

Задача. При осуществлении прохода через считыватель СКД «Аполло», показывать в течение 10 секунд диалоговое окно оператора (visitor.dlg) с отображением информации из базы данных: Идентификатор пользователя, ФИО, Год и место рождения, Сведения о документе, Сведения о прописке, К какому сотруднику пришел, Уровень допуска и Срок действия карточки.

Примечание: Идентификатор пользователя известен при получении события (param1), ФИО берется из базы с помощью функции GetObjectName, к какому сотруднику пришел, Уровень допуска и Срок действия карточки берутся из соответствующих полей в базе (person, level_id, expired) с помощью функции GetObjectParam. Далее, в зависимости от значения Уровеня допуска берется либо его имя либо присваивается одно из стандартных значений (Полный, Запрещен). Параметры Год и место рождения, Сведения о документе, Сведения о прописке берутся из одного поля comment (идут через разделитель "|"), и разделяются далее с помощью функции extract_substr. Переменная flag служит для того чтобы закрытие диалогового окна происходило только через 10 секунд после последней сработки считывателя , т.к. окно автоматически зарывается если поступает новое событие до истечения 10 секунд.

```
OnEvent("AAM_READER",N,"ACCESS_IN")

{[

flag=param1;

idn=param1;

fio=GetObjectName("PERSON",idn);

gmr=extract_substr(GetObjectParam("PERSON",idn,"comment"),"|",0);

doc=extract_substr(GetObjectParam("PERSON",idn,"comment"),"|",1);

prp=extract_substr(GetObjectParam("PERSON",idn,"comment"),"|",2);

sot=GetObjectParam("PERSON",idn,"person");

dat=GetObjectParam("PERSON",idn,"expired");

lev=GetObjectParam("PERSON",idn,"level_id");

DoReact("DIALOG","visitor","CLOSE_ALL");

if(strequal(lev,""))
{lev=GetObjectParam("DEPARTMENT",GetObjectParam("PERSON",idn,"parent
_id"),"level_id");}

if(strequal(lev,"-")) {lev="Доступ запрещен";}

else

{if(strequal(lev,"*")) {lev="Полный доступ";}

else

{if(strequal(lev,"")) {lev="Доступ запрещен";}
```

```
else

{lev=GetObjectName("LEVEL",lev);}}}

DoReact("DIALOG","visitor","RUN","idn<"+idn+">,fio<"+fio+">,gmr<"+gmr
+">,doc<"+doc+">,prp<"+prp+">,sot<"+sot+">,dat<"+dat+">,lev<"+lev+">");

Wait (10);

if(strequal(flag,param1)) {DoReact("DIALOG","visitor","CLOSE_ALL");}

]}
```

Пример 11

Микрофон (OLXA_LINE) пишется не синхронно с камерой. По умолчанию микрофон не стоит на охране.

Задача. 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 12

Есть определенное количество камер (num).

Задача. 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 13

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 милисекунд, 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

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

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" - устанавливает параметры платы видеоввода	chan<>	номер канала (0,1,2,…)
	mode<>	скорость оцифровки (0 – максимальная, 1 – средняя, 2 – минимальная)
	resolution<>	разрешение (0– четверть кадра (384x288), 1 – полукадр (768x288), 2 – кадр (768x576))
	format<>	формат сигнала (PAL, NTSC)
	drives<>	диски для записи видеоархива (DRIVE1:\, DRIVE2:\, … DRIVEN:\)
	cams<>	количество подключенных камер
"SET_DRIVES" - устанавливает диски для записи видеоархива	drives<>	диски для записи видеоархива
"MUX1_OFF" - отключить аналоговый выход 3	-	-
"MUX2_OFF" - отключить аналоговый выход 2	-	-

"MUX3_OFF" - отключить аналоговый выход 3	-	-

Пример:

DoReact("GRABBER", "1", "SETUP", "chan<1>,mode<0>,resolution<1>,format<PAL>");

«1-я плата видеоввода, канал – 1, скорость оцифровки – максимальная, разрешение – полукадр, формат – PAL»

Пример:

DoReact("GRABBER", "1", "SET_DRIVES", "drives<D:\,F:\>");

«Записывать видеоархив на диски D:\ и F:\»

Команды

"MUX1_OFF"

"MUX2_OFF"

"MUX3_OFF" отключают вывод видео через соответственно 1, 2, 3 аналоговые выходы граббера.

Пример:

// выведем камеру 1 на 1-ый аналоговый вывод платы

DoReact("CAM","1","MUX1");

Wait(5);

// отключаем 1-ый аналоговый выход 1 и 2 ой плат

DoReact("GRABBER","1","MUX1_OFF");

DoReact("GRABBER","2","MUX1_OFF");

Примечание: если аналоговые выходы 2-ух и более плат соединяются параллельно и камера 1 ,например, принадлежит первому грабберу, а камера 2 - второму, то при вызове команды

Примечание. Описание объекта "CAM" указано ниже (см. пункт "2.7.2. CAM").

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.

3.7.2 CAM

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

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" - устанавливает (изменяет) параметры камеры	rec_priority<>	приоритет записи (0 – обычный, 3 – все ресурсы)
	compression<>	степень компрессии (0 – компрессия отсутствует, 1- макс. качество, …, 5 – мин. качество)
	sat_u<>	уровень цветности (0 – мин, 10 – макс)
	proc_time<>	период дозаписи, с
	hot_rec_period<>	период горячей записи, мс
	manual<>	ручная установка уровней яркости и контрастности (0 – выключено, 1 – включено)
	telemetry_id<>	идентификатор модуля телеметрии (id поворотника)
	contrast<>	контрастность (0 – мин, 10 – макс)
	md_size<>	детектор движения – размер (0 – max, 10 – min)
	md_mode<>	режим записи пауз (1 – включено, 0 выключено)
	audio_type<>	тип звукового сопровождения
	pre_rec_time<>	период отката, с
	bright<>	яркость (0 – мин, 10 – макс)
	audio_id<>	номер микрофона (пустой параметр, если нет микрофона)

Команда - описание команды	Параметры	Описание параметров
	rec_time<>	период записи, мс
	alarm_rec<>	запись тревог (1 – включено, 2 – выключено)
	hot_rec_time<>	время горячей записи, с
	hot_rec_period<>	период горячей записи, мс
	rec_time<>	период записи, мс
	md_mode<>	режим записи пауз (1 – включено, 0 выключено)
	mux<>	номер канала (0 – 1 канал, 15 – 16 канал)
	color<>	цветность (0 – выключено, 1 – включена)
"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 аналоговый 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

Примеры:

DoReact("CAM","1","SETUP","rec_priority<2>"); - использовать половину ресурсов при записи, то есть, если в системе через 1 граббер подключено 4 камеры, то 1 – ая будет записывать 6 кадров/сек, а остальные три - по 2 – 2,5 кадра/сек.

DoReact("CAM","1","SETUP"," priority<2>"); - использовать половину ресурсов при отображении, то есть, если в системе через 1 граббер подключено 4 камеры, то 1 – ая будет отображать со скоростью 6 кадров/сек, а остальные три - по 2 – 2,5 кадра/сек.

DoReact("CAM", "1", "SETUP", "compression<5>, audio_type<OLXA_LINE>, audio_id<4>"); 1-я камера, максимальная компрессия, синхронно с 4-м микрофоном звуковой платы.

value = 5;

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

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

DoReact("GRABBER", video_canal_id, "SETUP", "chan<0>,mode<0>,resolution<1>,format<PAL>"); определяем идентификатор видеоканала, которому принадлежит камера 1, и устанавливаем новые параметры видеоканала.

3.7.3 MONITOR

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

Общие замечания: slave_id – имя компьютера, которому принадлежит монитор, в скрипте можно подставить owner.

control – 0 только просмотр архива, 1 – так же возможно и управление (постановка/снятие с охраны, запись).

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 корой необходимо остановить
"REPLACE" - удаляет все камеры с монитора и вызывает указанную камеру	slave_id<>	-
	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<>	-
"ADD_SHOW**"** -	cam<>	id камеры в дереве настроек, которую необходимо вывести на монитор
	name<>	название камеры, которое будет отображаться в левом нижнем углу
	arch_id<>	-
	control<>	-
"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<>	-
	password<>	-
	overlay<>	-

Команда - описание команды	Параметры	Описание параметров
	x<>	-
	y<>	-
	w<>	-
	h<>	-
	max_cams<>	-
	min_cams<>	-
	compress<>	-
	panel<>	-
	panel_type<>	-
	s<>	-
	layout<>	-
	gate<>	-
	map_id<>	-
	enable<>	-
	topmost<>	1 - показывать экран поверх всех остальных окон
"ACTIVATE" - активирование панели управления монитора	user_id<>	-
	panel_active<>	-
"DEACTIVATE" - де активирование панели управления монитора	-	-
"EXPORT_FRAME" - экспорт кадра в JPG-файл	cam<>	-
	file	-
"KEY_PRESSED"	key<>	"ARCH_EDIT_DATE" "ARCH_EDIT_TIME" "ARCH_EDIT_ENTER" "ARCH_EDIT_ESCAPE" "ARCH_EDIT_BACK" "ARCH_EDIT_REPLACE" "WINDOW_ZOOM_IN"

Команда - описание команды	Параметры	Описание параметров
		"WINDOW_ZOOM_OUT" "ZOOM_IN" "ZOOM_OUT" "CYCLE_REW" "CYCLE_FF" "LEFT" "RIGHT" "UP" "DOWN" "MODE_VIDEO" "MODE_ARCH" "MODE_ARCH2" "MASK_SHOW" "MASK_HIDE" "ARM" "DISARM" "REW" "PLAY" "PLAY_NONSTOP" "PLAY_FAST" "FF" "RECORD" "RECORD_MIC" "STOP" "REC_STOP" "PAUSE" "MIC_ON" "MIC_OFF" "PRINT"

Команда - описание команды	Параметры	Описание параметров
	number<>	-

Примеры:

// проиграть запись с камеры 1 на мониторе 4 с указанными датой и временем:

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

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

// перейти в режим просмотра видеоархива на 1-ой камере монитора 4, и перейти на 10 кадров далее, начиная с фрагмента указанной даты и времени

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>");

}

3.7.4 AUDIO

Формат: DoReact("AUDIO","_id_","_команда_" ["_параметры_"]);

Команда - описание команды	Параметры	Описание параметров
"PLAY_WAV" - проигрывает звуковой файл	file<>	звуковой файл с полным путем к нему
"RECORD_START" - включает запись звука с микрофона	-	-
"RECORD_STOP" - выключает запись звука с микрофона	-	-

"PLAY_START" - проигрывание аудиозаписи	-	-
"PLAY_STOP" - остановка проигрывания аудиозаписи	-	-

3.7.5 DIALOG

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

Формат: DoReact("DIALOG","_id_","_команда_" [,"_параметры_"]);

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" - настройка окна запроса оператора	x<>, y<>, allow_move<>	x,y - координаты вывода, allow_move – 0 – запретить перемещение, 1 – разрешить перемещение
"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	-	-

Пример 1. По макрокоманде с номером 1 устанавливать координаты верхнего левого угла окна запроса оператора поворотной камеры 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");

}
```

Пример 2. По макрокоманде с номером 2 закрывать окно запроса оператора.

```
OnEvent("MACRO","2","RUN")
{
DoReact("DIALOG","panasonic-850","CLOSE");
}
```

3.7.6 MMS

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

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" - настройки для отправления почтовых сообщений	smtp<>	IP-адрес сервера
	connection<>	имя соединения
	username<>	имя пользователя
	password<>	пароль

3.7.7 MAIL_MESSAGE

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

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" - настройки для почтового сообщения	from<>	адрес почтового ящика, откуда исходит сообщение
	to<>	адрес почтового ящика, куда отправить сообщение
	cc<>	адрес почтового ящика, куда отправить копию сообщения
	bcc<>	адрес почтового ящика, куда отправить слепую копию сообщения
	subject<>	тема письма
	body<>	тело письма
	attachments<>	прикрепленный файл

"SEND" – отправка почтового сообщения	-	-

Пример скрипта отправки сообщений при срабатывании датчика движения вместе с картинкой от камеры:

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

{

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");

}
```

3.7.8 VDIAL

Формат: DoReact("VDIAL","_id_","_команда_" [,"_параметры_"]);

Команда - описание команды	**Параметры**	**Описание параметров**
"SEND" - послать звуковое сообщение по телефону	port<>	номер COM-порта
	phone<>	номер телефона (пример 9W;8W;;0959980505 W - ждать длинного гудка, ; - пауза 2 сек)
	pulse<>	тип набора, 0 - тоновый, 1 - импульсный
	sens<>	чувствительность, от 0 до 10
	skip<>	задержка соединения после набора, 300 - для обычных линий, 5000-7000 - для сотовых
	file<>	имя файла для проигрывания (в звуковом формате .wav)
	while_send<>	0 - в случае если номер не отвечает прекратить дозвон, 1 - дозваниваться пока не будет голосового ответа (использовать с осторожностью - может надолго занять телефонную линию)

Пример: (модем подключен к порту COM2)

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

{

DoReact("VDIAL","1","SEND","port<2>,phone<;8;;
;9021600212>,pulse<1>,file<f:\Inspector\wav\test.wav>,while_send<0>,sens<6>,skip<5
00>");

}
```

3.7.9 RELE

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

Команда - описание команды	**Параметры**	**Описание параметров**
"ON" - включить реле	-	-
"OFF" - выключить реле	-	-

3.7.10 RAY

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

Команда - описание команды	**Параметры**	**Описание параметров**
"ARM" - поставить на охрану луч	-	-
"DISARM" - снять с охраны луч	-	-

3.7.11 VNS

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

Команда - описание команды	**Параметры**	**Описание параметров**
"SETUP" – настройка голосового оповещения	card<>	Название звуковой карты. Примечание: Имя карты должно строго соответствовать тому названию, что указано в настройках звуковой карты «Сервиса голосового оповещения» системы «Интеллект».
	level<>	Значение регулятора громкости. Значение параметра варьируется от 0 до 15. По умолчанию

Команда - описание команды	Параметры	Описание параметров
		оно равно 8, то есть среднему.
	channel<>	Канал воспроизведения. Возможные значения параметра: 0 – нет звукового канала; 1 – левый канал воспроизведения; 2 – правый канал воспроизведения; 3 – левый и правый канал воспроизведения (оба канала).
"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 «+».

Пример кода голосового оповещения, реагирующего на событие - начало записи камеры:

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

{

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

}
```

Пример. При наступлении, заранее заданной временной зоны, данный код меняет значение регулятора громкости, затем по её окончании, ставит 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>");

}
```