Проект Anima

Предварительное описание. Версия 2

Сергей Гурин, сентябрь 2009

Введение

Проект Anima – это продолжение и развитие проекта Visual2k. Anima имеет то же назначение, что и Visual2k, но позволяет разрабатывать программы со значительно большими возможностями, чем свой предшественник. Anima использует современную элементную базу – тридцатидвухразрядные микроконтроллеры и современную операционную технологию – Microsoft .NET. В отличие от Visual2k проект Anima ориентирован на программирование не только автономных кукол, но и на создание сложных кукольных спектаклей, которые объединяются общим сценарием, включают большое число кукол и компьютеров. Технологическая основа объединения разнородных микроконтроллеров и компьютеров – локальная сеть, в которой отдельные узлы децентрализованно взаимодействуют друг с другом.

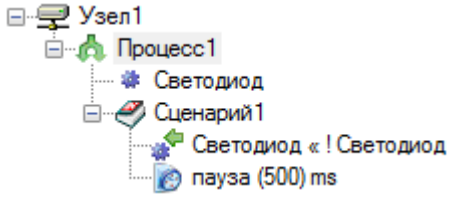
При разработке Anima особо обращалось внимание на следующие моменты:

1. Модульная развиваемая структура, которая позволяет относительно легко добавлять новые типы микроконтроллеров, компьютеров, приводов, датчиков, сетевых протоколов;
2. Возможность создавать достаточно сложные и гибкие визуальные интерфейсы взаимодействия со зрителями, используя мультимедийные возможности современных персональных компьютеров;
3. Лицензионная чистота программного продукта.

Первая программа

Для демонстрации того, как происходит создание Anima-программ, подробно рассмотрим очень простой пример и объясним некоторые основные элементы языка программирования и интегрированной среды проектирования. Задача, которая будет решаться в этом примере – управление простым устройством с двумя состояниями. Аппаратная основа примера – автономный микроконтроллер LPC2378 и светодиод (используется отладочная плата LPC-2378-STK). Сценарий очень прост – светодиод должен мигать с заданным периодом. Все демонстрационные программы примеров находятся в каталоге Projects\Примеры.

Пример1.



Anima-программа похожа на Visual2k-программу и выглядит как дерево графических вложенных объектов. Отдельные уровни-веточки программы можно скрывать и раскрывать, делая доступным тот или иной фрагмент программы. Рассмотрим подробнее отдельные операторы программы.

Узел

Поскольку наш пример использует единственный автономный микроконтроллер, то узел всего один. Более сложные проекты будут включать множество узлов. Как правило, узлы одного проекта связаны между собой локальной сетью, но в рамках одного проекта могут также сосуществовать как автономные узлы, так и группы узлов, связанных между собой, но не связанных с остальными узлами. Узел, как и любой другой объект программы, имеет свойства-атрибуты:

1. Общие атрибуты	
Имя узла	**Узел1**
Описание	
Атрибуты узла	
Процессор	**LPC2378**
Шлюз	
DMX512	

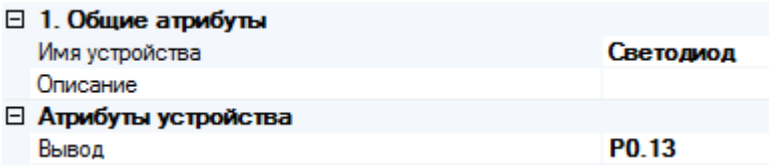
Из атрибутов узла особо отметим тип процессора. В примере используется микроконтроллер LPC2378 с системой команд ARM7 компании Philips (NXP). Второй атрибут, тип шлюза, определяет способ соединения узла с другими узлами сети. В нашем случае такого соединения нет. Третий атрибут определяет свойства контроллера DMX512. В нашем примере контроллер DMX не используется.

Параллельный процесс

Вся активная работа, которую выполняет узел, разделяется на ряд параллельных процессов. Каждый процесс полностью автономен и независим от других процессов узла. Забегая вперёд отметим, что параллельные процессы могут взаимодействовать друг с другом с помощью каналов, причем неважно, работают ли процессы в одном узле или в разных узлах сети. В нашем примере достаточно только одного параллельного процесса.

Устройство

Каждый процесс обладает своим собственным набором вложенных объектов, в данном случае есть только один объект – устройство (светодиод). Светодиод относится к устройствам-коммутаторам и имеет следующий набор атрибутов:



Устройства-коммутаторы имеют один логический сигнал и два состояния – активное (включенное) и неактивное (выключенное). Включенное состояние соответствует нулевому (активному) уровню на выводе микроконтроллера, выключенное состояние соответствует ненулевому (неактивному) уровню. Активный уровень вызывает включение светодиода, неактивный — выключение. Атрибут устройства определяет вывод микроконтроллера, к которому подключен светодиод, в нашем случае это вывод P0.13.

Сценарий

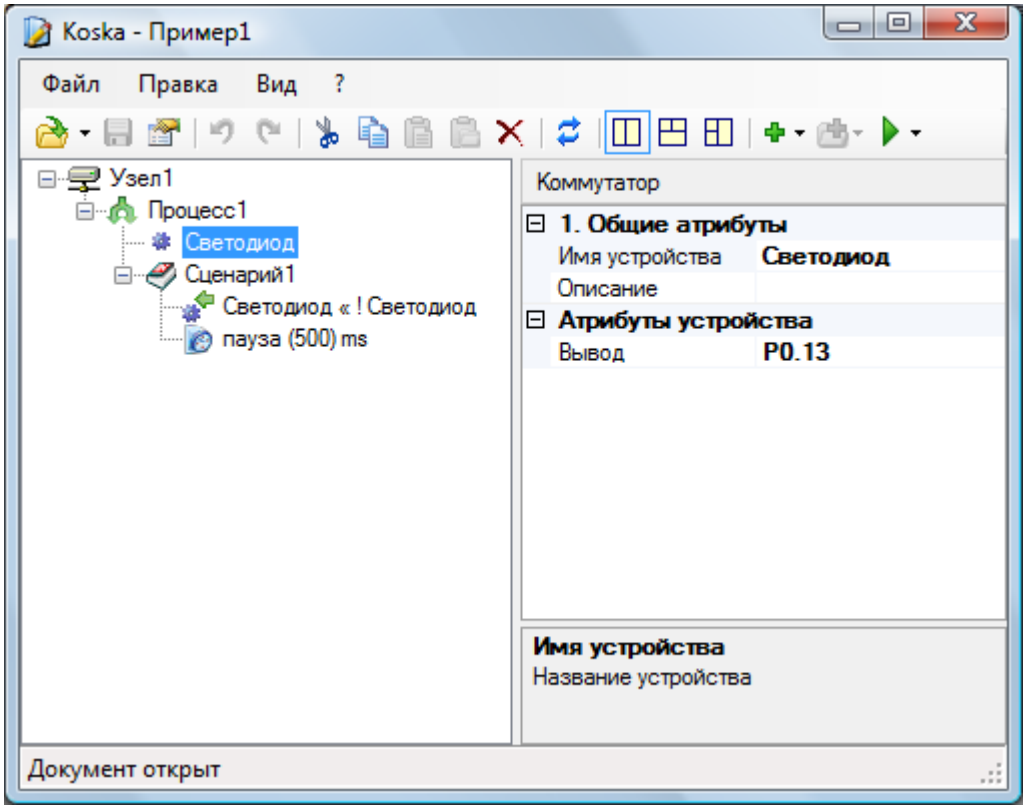
Работа параллельного процесса разбивается на ряд мелких фрагментов – сценариев. Сценарии можно рассматривать как состояния процесса. Каждый сценарий циклически выполняется до тех пор, пока явно не перейдёт к другому сценарию. Если такого перехода нет, то сценарий будет выполняться бесконечно. В нашем примере только один сценарий, который имеет всего два действия — изменяет состояние светодиода на противоположное и выдерживает паузу.

Первый оператор сценария изменяет состояние светодиода на противоположное. Оператор выполняет посылку значения в устройство. В нашем случае, в устройство-светодиод посылается противоположное значение самого светодиода. Операция ! (логическое «НЕ») меняет значение операнда на противоположное, то есть, если значение светодиода было ложным (нулевым, неактивным), то оно становится истинным (ненулевым, активным) и наоборот. Значение, передаваемое в устройство, может быть задано произвольным арифметическим выражением. При старте программы состояние всех устройств устанавливается неактивным.

Второй оператор сценария выдерживает паузу, равную 500 миллисекундам. Интервал времени задается в микросекундах, миллисекундах или секундах. Значение интервала может быть числом или произвольным арифметическим выражением.

Интегрированная среда проектирования

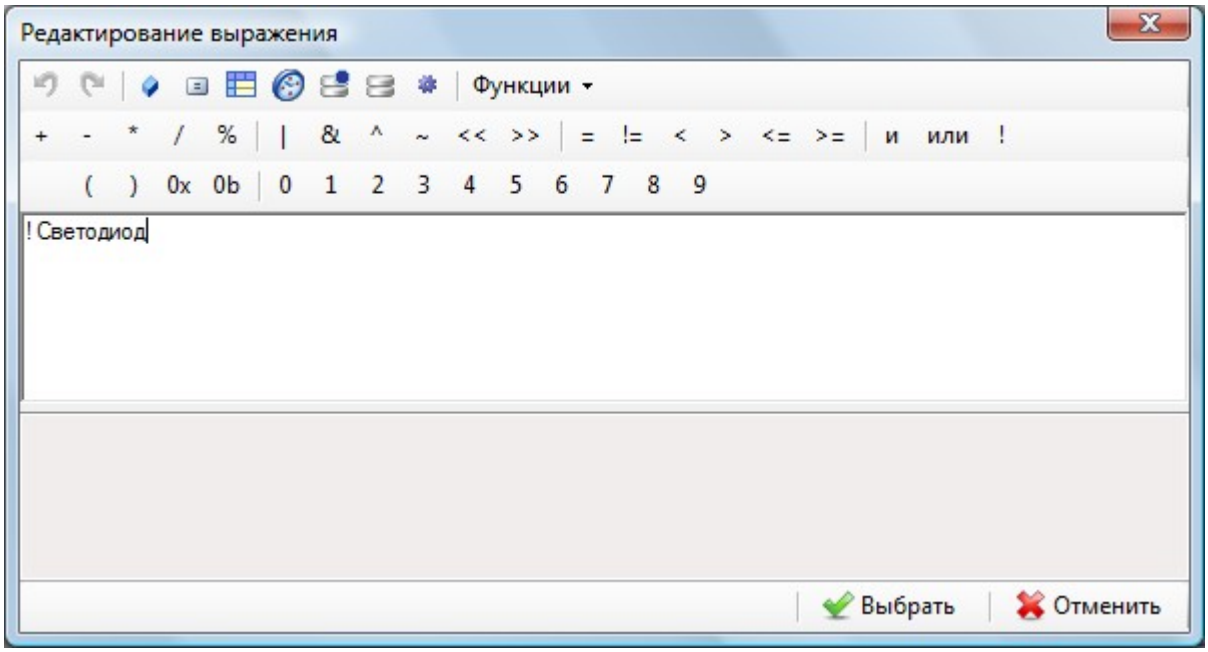
Теперь вкратце рассмотрим интегрированную среду, в которой разрабатывается проект. В Anima интегрированная среда представлена программой KoskaXmlEditor. Это инструмент универсального назначения, который был создан мною как результат собственных исследований и используется в различных проектах для различных заказчиков. Специфика Anima задается специальным модулем, подключённым к программе KoskaXmlEditor.



Окно программы содержит меню, полоску кнопок и рабочую область, которая разбита на две части – область программы (слева) и инспектор объектов (справа).

Инспектор объектов

Инспектор имеет заголовок – тип инспектируемого объекта, список атрибутов (свойств), сгруппированных по категориям и поле подсказки, которое поясняет смысл каждого атрибута. Атрибут может иметь список возможных значений или мастер ввода значений. Например, атрибут «Значение» можно просто ввести в строке ввода значения, или перейти к мастеру, щелкнув по маленькой кнопке справа от строки ввода значения:

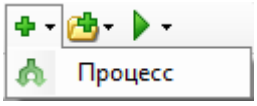


Различные атрибуты имеют свои собственные мастера, облегчающие выбор значений.

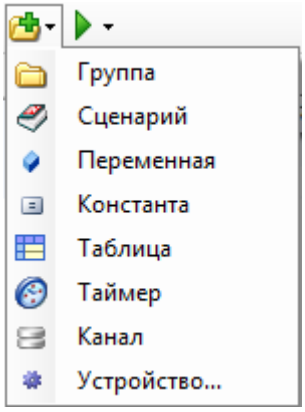
Область программы

Теперь покинем инспектор и перейдём в левую часть рабочей области. Для добавления операторов имеется две кнопки:

Первая кнопка добавляет новый оператор за текущим оператором, а вторая – внутрь текущего оператора. При нажатии на любую из этих кнопок появляется меню, содержащее список операторов, допустимых в данной точке программы. Например, если текущий оператор – процесс, то при щелчке по первой кнопке будет доступно добавление только процесса:

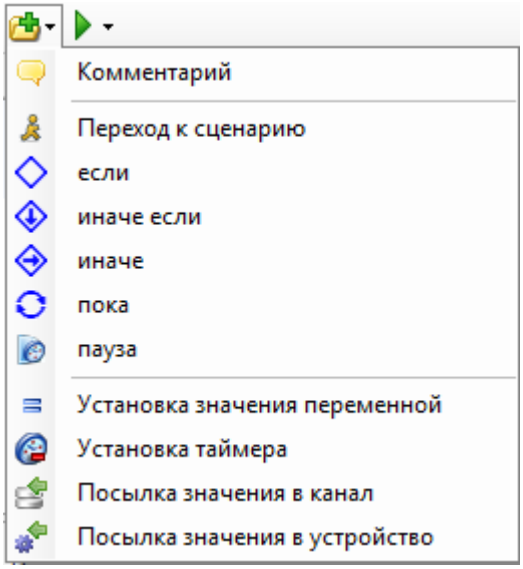


Это означает, что узел может содержать только процессы и ничего больше. Но при щелчке по второй кнопке меню будет другим:



Это означает, что процесс может содержать вложенные объекты типа «группа», «сценарий», «переменная», «константа» и т. д.

Сценарий, в свою очередь, может содержать такие вложенные объекты-операторы:



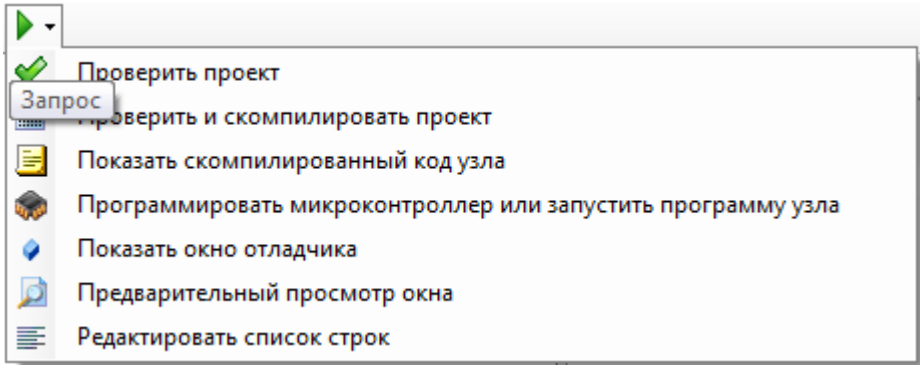
Редактирование программы

При редактировании используются обычные операции отмены и возврата, копирования и вставки из буфера обмена или удаления.

Отмена и возврат действуют на создание, удаление, перемещение объектов и на изменение их атрибутов. Объекты копируются и вставляются вместе со всеми своими вложенными объектами. Для перемещения объектов можно использовать буксировку мышкой (перетащить и бросить).

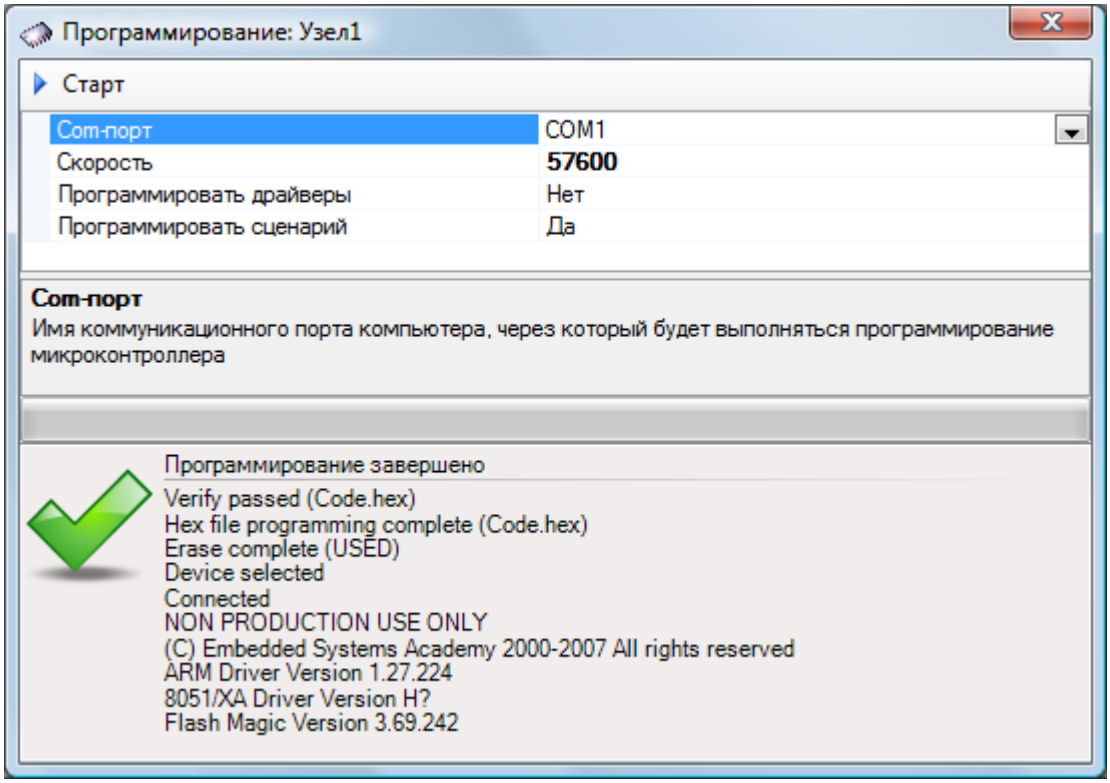
Запросы

После того, как программа создана, с ней можно делать некоторые действия-запросы, щелкая по последней кнопке в полоске кнопок:



Наиболее важны два запроса – «Проверить и скомпилировать проект» и «Программировать микроконтроллер или запустить программу узла». Результат компиляции не зависит от типа микроконтроллера или компьютера. Скомпилированная программа представляет собой байт-код для виртуальной машины, подобно тому как это делается в языке Java. Такая виртуальная машина вместе с драйверами устройств разрабатывается для каждого типа микроконтроллера (или компьютера), который поддерживается в Anima.

Второй запрос «Программировать микроконтроллер или запустить программу узла» либо программирует микроконтроллер, либо запускает программу на персональном компьютере. Программирование заключается в копировании скомпилированной программы во внутреннюю flash-память микроконтроллера. Для этого используются различные средства. Например, программирование LPC2378 выполняется свободно распространяемой утилитой FlashMagic через Com-порт. Для микроконтроллеров LPC2378 мастер запроса «Программировать микроконтроллер узла» выглядит так:



Операция зашивки программы в микроконтроллер – завершающая, после неё микроконтроллер начнёт выполнять созданную нами программу и мигать светодиодом.

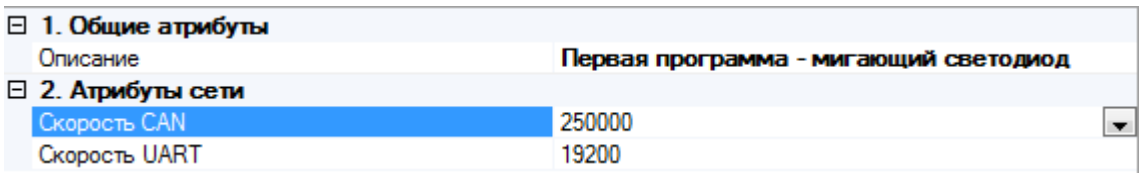
Операторы языка Anima

В этом разделе будут описаны все операторы языка. Особенности, зависящие от типа процессора, вынесены в последующие разделы.

Проект

Проект — это корень и контейнер для всех остальных операторов Anima-программы. Для создания нового проекта используется команда соответствующая команда меню интегрированной среды KoskaXmlEditor.

После создания нового пустого проекта инспектор объектов будет отображать свойства проекта:



Свойства проекта:

Описание	Необязательный комментарий, описывающий назначение и особенности проекта
Скорость CAN	Скорость передачи данных между узлами сети по каналам CAN (в бодах). Допустимые значения: 50000, 125000, 250000, 500000. Скорость зависит от максимальной длины кабеля всей сети. При скорости 500000 длина кабеля не должна превышать 50 метров, при скорости 50000 длина кабеля может быть до километра. Кроме того, максимальная скорость также зависит и от качества кабеля, поэтому целесообразно вначале использовать относительно небольшую скорость и увеличивать её при хорошей, устойчивой связи
Скорость UART	Скорость передачи данных между узлами, соединёнными по каналам Com-UART (в бодах). Допустимые значения: 9600, 19200, 38400, 57600, 115200. Скорость зависит от длины и качества кабеля. Для скорости 115200 длина кабеля не должна превышать один-полтора метра, а для скорости 9600 – десяти метров

Свойства проекта всегда можно посмотреть с помощью команды меню «Свойства корневого объекта». После создания нового проекта его нужно сохранить с помощью команды меню «Сохранить как...». Рекомендуется сохранять все проекты в каталоге Projects, который находится в основном каталоге Anima. Каждый проект должен иметь в Projects свой каталог и файл программы проекта. Рекомендуется также сохранять файл программы под тем же именем, что и каталог проекта. Если несколько проектов связаны между собой общей целью, то можно создать специальный общий подкаталог. В качестве примера можно привести файловую структуру группы проектов «Примеры». Исходный файл программы имеет расширение xml. Проект может содержать один или несколько узлов.

Узел

Узел определяет компонент сети. Свойства узла:

Имя узла	Уникальное имя узла в сети. Именно это имя будет иметь скомпилированная программа узла. Имя узла должно быть

	идентификатором, то есть, оно может содержать буквы, цифры и знак подчерка _. Имя должно начинаться с буквы
Описание	Необязательный комментарий
Процессор	Один из списка процессоров, которые поддерживаются в Anima. Полный список процессоров и их особенностей будет представлен в одном из последующих разделов. Процессор нужно определить в первую очередь, так как значение остальных атрибутов зависит от типа процессора
Шлюз	Один из списка шлюзов, которые поддерживаются в Anima. Список шлюзов и возможные топологии сети будут подробно описаны ниже. Различные процессоры имеют различные шлюзы. Если узел не связан сетью с другими узлами, то тип шлюза нужно оставить пустым.
DMX512	Если узел поддерживает взаимодействие с устройствами стандарта DMX512, то в атрибуте нужно указать тип процессорного канала, который будет управлять линией DMX512. Если узел не связан с DMX512, то тип канала нужно оставить пустым. В настоящее время предусмотрен только однонаправленный канал DMX512 – от микроконтроллера к устройствам-фонарям. То есть, микроконтроллер может только формировать и передавать пакет DMX512, но не принимать его. Приемный DMX512 может быть разработан в будущем, если потребуется поддерживать стандартные, фабрично изготовленные устройства, формирующие пакет DMX512.

При компиляции проекта для каждого узла будет создано по два файла. Имена обоих файлов совпадают с именем узла. Файл скомпилированной программы имеет расширение bin, а отладочный файл имеет расширение dbg. bin-файл используется для зашивки в микроконтроллер или для запуска на компьютере, а dbg-файл используется отладчиком.

Узел может содержать один или несколько параллельных процессов.

Процесс

Параллельный процесс – это единица работы узла. Все процессы выполняются параллельно и независимо друг от друга. Вложенные объекты, которыми обладает процесс, доступны только ему и недоступны остальным процессам. Взаимодействие параллельных процессов выполняется с помощью каналов. Атрибуты процесса:

Имя	Имя процесса однозначно определяет процесс в узле. Процессы должны иметь уникальные имена в пределах узла, но различные узлы может содержать процессы с одинаковыми именами. Имя процесса должно быть допустимым идентификатором
Описание	Необязательный комментарий к процессу

Процесс может содержать следующие вложенные объекты:

- Группа
- Сценарий
- Переменная
- Константа
- Таблица
- Таймер
- Канал
- Устройство

Группа

В отличие от других объектов, группа не является значимым объектом и ничего не делает, кроме того, что группирует другие, значимые объекты. Если процесс имеет немного вложенных объектов, то группировать их не имеет смысла. Если же процесс имеет большое число вложенных объектов, то для удобства просмотра программы можно сгруппировать объекты по некоторому критерию, например, создать отдельную группу для переменных, отдельную группу для каналов и так далее. Группа может содержать те же самые вложенные объекты, что и процесс. Кроме того, группы могут быть вложены друг в друга. Атрибуты группы:

Имя	Имя группы в процессе. В отличие от других объектов, имя группы не обязано быть идентификатором – оно может содержать любые символы, включая пробелы
Описание	Необязательный комментарий, характеризующий группу

Переменная

Все Anima-переменные имеют целый тип и могут хранить целое число со знаком в диапазоне от -4294967295 до +4294967295. Переменная локальна по отношению к процессу, доступ к переменной разрешён только тому процессу, в котором переменная определена. Никакой другой процесс не может получить или установить значение переменной. Значение переменной устанавливается оператором присваивания (оператором установки значения переменной). Значение переменной может быть использовано в различных операторах процесса. Атрибуты переменной:

Имя	Имя переменной должно быть уникальным в пределах процесса, то есть, наличие в процессе двух переменных с одинаковым именем недопустимо. Имя должно быть идентификатором
Описание	Необязательный комментарий, характеризующий назначение переменной

Константа

В отличие от переменной, значение константы задаётся один раз и не может быть изменено. Так же, как и переменная, константа имеет целый тип. Значение константы находится в диапазоне от -4294967295 до +4294967295. Поскольку значение константы вычисляется в момент компиляции программы, константа может быть безопасно доступна всем узлам Anima-проекта. Это единственное исключение из правила, по которому внутренние объекты процесса не могут быть доступны другим процессам. Каналы – это особый случай, так как их назначение как раз и состоит во взаимодействии между различными процессами. Атрибуты константы:

Имя	Имя константы должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение константы
Значение	Число или константное выражение. Выражение может содержать числа, константы и использовать все арифметические операторы. Выражение не может содержать изменяемых объектов программы – переменных, функций, каналов и т. д. Недопустимо рекурсивное определение констант, при котором значения двух и более констант взаимно зависят друг от друга.

Таймер

Таймер предназначен для отсчёта интервалов времени. Процесс может содержать множество таймеров, используемых для разных целей. Обычно таймер используется для реализации паузы или таймаута какой-то операции. Атрибуты таймера:

Имя	Имя таймера должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение таймера

Канал

Канал является средством коммуникации (взаимодействия) параллельных процессов, причем неважно, работают ли процессы в одном узле или в разных узлах сети. Канал связывает однонаправленной связью только два процесса – процесс-отправитель и процесс-получатель. Читать состояние канала может только один процесс, которому канал принадлежит, то есть, тот процесс, в котором канал определён. Канал не может быть использован для множественного доступа, то есть, невозможно использовать канал для передачи данных от нескольких процессов к одному или от одного процесса к нескольким процессам. Если требуется двунаправленное взаимодействие, то необходимо создать два канала для обоих направлений передачи. Данные, передающиеся по каналу, представлены целым числом (в диапазоне от -4294967295 до +4294967295). Канал представляет собой очередь сообщений с единичной длиной и имеет два свойства — готовность и значение. Готовность свидетельствует о том, содержит ли канал принятое значение или нет. Значение канала можно получать только после проверки его готовности. Признак готовности автоматически сбрасывается после его проверки. Например, если в канал было послано одно данное, то первая проверка готовности канала будет давать истинный результат, а последующие — ложный. Значение канала можно читать многократно — оно не изменяется вплоть до посылки следующего данного. Атрибуты канала:

Имя	Имя канала должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение канала
Отправитель	Имя процесса, которому разрешено передавать данные по каналу. Только указанный процесс может послать данные в канал — попытка посылки в канал другим процессом будет воспринята как ошибка на этапе компиляции программы

Взаимодействие по каналу принципиально ненадежно — для этого может быть множество причин, например, сбой или обрыв связи между узлами или переполнение очередей сетевых драйверов. Как бы то ни было, любая причина приведет в конечном итоге к тому, что сообщение по каналу не будет получено. Поэтому те каналы, которые требуют надежной доставки сообщений, должны поддерживать какой-либо алгоритм квитанции — подтверждение получения сообщения. Если квитанция не получена за некоторое предельно допустимое время, то процесс-отправитель должен повторить посылку в канал. Если квитанция не получена после нескольких попыток, это свидетельствует о неустранимой причине, например, обрыве кабеля и требует перехода к альтернативному алгоритму работы. Пример организации надежной связи по каналу показан в проекте «Пример2».

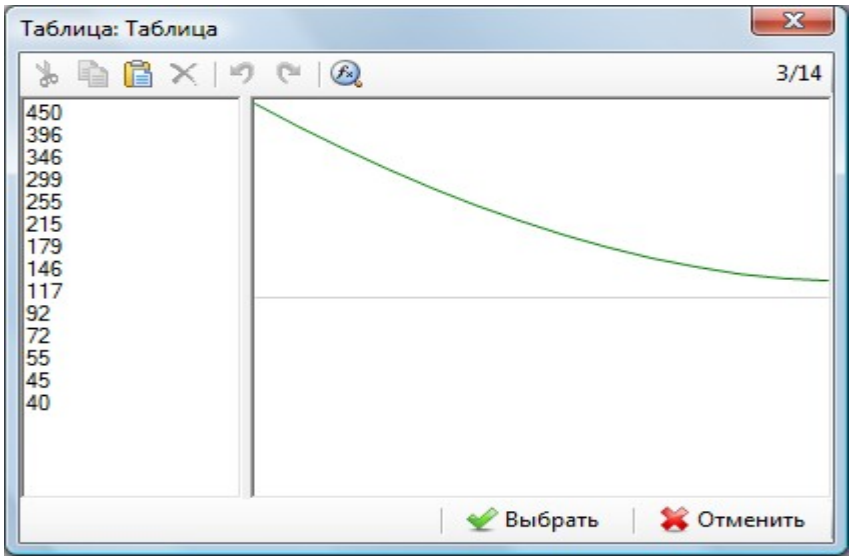
Важное замечание. Надежность каналов существенно зависит от типа шлюза. Существуют шлюзы с большей и меньшей надежностью. Например, шлюз CAN характеризуется очень высокой степенью надежности, так как на аппаратном уровне контролирует правильность передачи и автоматически делает многократные попытки повторения передачи при любом

типе ошибок, так что в большинстве случаев для CAN не требуются дополнительные программные средства обеспечения надежности. Шлюз RS-232 предоставляет менее надежные каналы, поэтому при высоких требованиях к надежности передачи для него нужен описанный выше алгоритм.

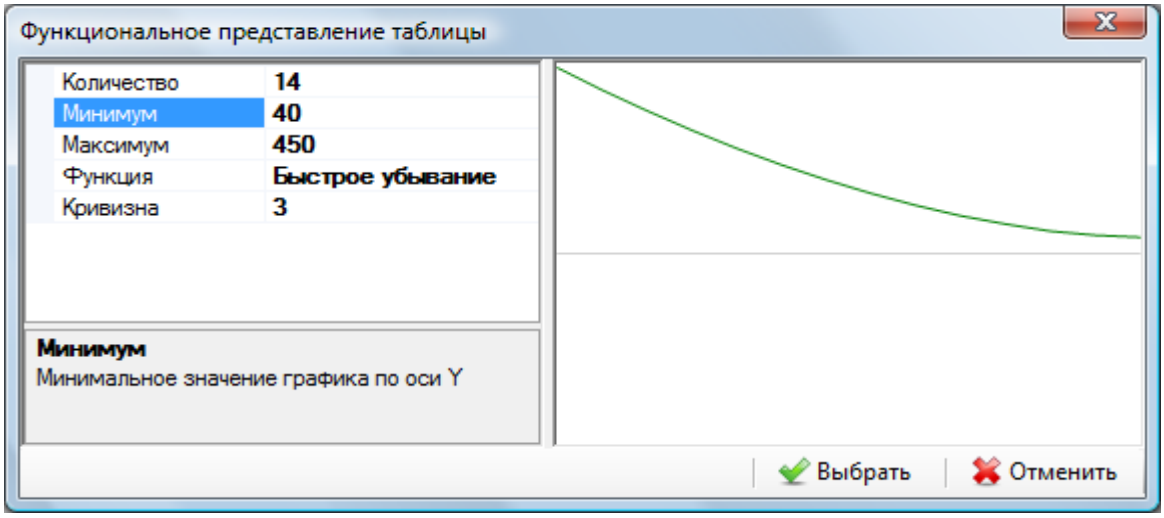
Таблица

Таблица представляет собой одномерный константный массив целых чисел. Элементы таблицы доступны по целому индексу от нуля до размера таблицы минус 1. Размер таблицы может быть любым. Таблица может быть использована в самых разных случаях, например, для задания нелинейной функции или последовательности аудио-файлов. Таблицу можно задавать в виде текстовой строки, разделяя значения символом ; (точка с запятой), например, 1;3;12;49;56. Но более удобно использовать мастер.

Мастер задания таблицы имеет два поля — поле задания значений и поле графика. Поле задания значений это обычный многострочный текстовый редактор, в котором можно использовать все операции текстового редактирования. Каждая строка должна содержать одно значение. Поле график отображает список значения в виде обычного графика. Два числа в правой части полоски кнопок показывают индекс текущего элемента и число элементов. Кроме операций текстового редактирования мастер имеет возможность задания значений таблицы как функции. Пример использования таблицы дан в проекте «Пример3». Мастер ввода таблицы для этого примера выглядит так:



Значения плавного нелинейного задания изменения были созданы операцией «Задать функцию» с помощью дополнительного мастера функций:



В функциональном представлении генерируются сразу все значения таблицы. Число значений и атрибуты функции определяются инспектором. Представление определяет несколько линейных и нелинейных функций.

Атрибуты таблицы:

Имя	Имя таблицы должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение таблицы
Данные	Массив данных, который можно задавать либо прямым вводом, либо с помощью мастера

Устройство

Под устройством понимается всё то, чем может управлять Anima-программа. Это может быть реальное устройство, например, мотор или фонарь, или виртуальное устройство, например, текст или графический образ на экране монитора. Набор устройств специфичен для конкретного процессора и будет дан в последующих разделах. Устройство может иметь одно или несколько состояний, свойств или действий. Набор свойств жёстко определён для каждого конкретного устройства. Например, устройство-мотор может иметь одно свойство, характеризующий скорость устройства. Если скорость не равна 0, то мотор вращается со скоростью, заданной свойством «Скорость» в том или ином направлении в зависимости от знака скорости. Если скорость равна нулю, то мотор остановлен. Другой пример — действие «Воспроизведение» для виртуального графического устройства (проигрывателя видео-аудио-файлов).

Мастер выбора устройства предоставляет полный список устройств, доступных для процессора, который реализует данный узел. Если тип процессора для узла не задан, то список будет пустым.

Сценарий

Сценарий это единица работа параллельного процесса. Сценарии можно рассматривать как состояния процесса. Каждый сценарий циклически выполняется до тех пор, пока явно не перейдёт к другому сценарию. Если такого перехода нет, то сценарий будет выполняться бесконечно.

Атрибуты сценария:

Имя	Уникальное имя сценария в процессе. В отличие от других объектов, имя сценария не обязано быть идентификатором – оно может содержать любые символы, включая пробелы
Описание	Необязательный комментарий, характеризующий сценарий

Сценарий может содержать следующие вложенные операторы:

- Комментарий
- Переход к сценарию
- Если
- Иначе если
- Иначе
- Пока
- Пауза
- Установка значения переменной
- Установка таймера
- Посылка значения в канал
- Посылка значения в устройство

Операторы Anima-программы

В этом разделе описываются операторы, которые может выполнять сценарий.

Комментарий

Комментарий не выполняет никаких значимых действий и используется только для комментирования программы. Атрибуты комментария:

Текст	Произвольный текст комментария

Переход к сценарию

Этот оператор завершает выполнение сценария и передает управление другому сценарию. Если сценарий не содержит операторов перехода, то он выполняется в бесконечном цикле. Атрибуты оператора:

Сценарий	Сценарий, к которому будет выполняться переход. Если сценарий не задан, то выполняется переход к следующему по-порядку сценарию. Мастер выбора сценария, доступный по маленькой кнопке справа от поля ввода имени предоставляет полный список всех сценариев процесса

Условный оператор «если»

Условный оператор проверяет значение логического выражения и выполняет вложенные операторы только в том случае, если значение выражения истинно (отлично от нуля). Условный оператор может содержать те же самые операторы, что и сценарий. Допускается вложение условных операторов на любую глубину. Атрибуты оператора:

Выражение	Условное выражение. Если выражение сложное, то его удобно вводить с помощью мастера

Условный оператор «иначе если»

Этот условный оператор может появляться только после предыдущего условного оператора «если» или «иначе если». Оператор «иначе если», как и оператор «если», имеет один атрибут — условное выражение. Из условных операторов можно строить цепочки проверки. Гарантируется, что при выполнении условия все остальные условные операторы в цепочке выполняться не будут.

Альтернативный условный оператор «иначе»

Этот оператор может появляться после условного оператора. Его вложенные операторы выполняются только в том случае, если значения предыдущих условных операторов ложны (равны нулю). Альтернативный условный оператор может содержать те же самые операторы, что и сценарий. Оператор не имеет атрибутов.

Оператор цикла «пока»

Поскольку сами сценарии цикличны, то потребность в операторе цикла возникает редко — только в том случае, когда внутри сценария нужно выполнить какое-то простое циклическое действие. Оператор подобен условному оператору с той разницей, что он выполняет вложенные операторы до тех пор, пока логическое условие истинно (неравно нулю).

Пауза

Оператор приостанавливает выполнение параллельного процесса на указанный интервал времени. Атрибуты оператора:

Значение	Число или произвольное арифметическое выражение. Если выражение сложное, то его удобно вводить с помощью мастера
Единица	Значение интервала времени можно задавать в секундах, миллисекундах или микросекундах. Отметим, что не все типы процессоров реально способны задавать интервал с микросекундной точностью

Оператор паузы можно заменить таймером и циклом.

Установка значения переменной

Этот оператор присваивает переменной новое значение. Атрибуты оператора:

Имя переменной	Имя переменной. Мастер выбора переменной предлагает полный список переменных, определённых в процессе
Значение	Число или произвольное арифметическое выражение. Если выражение сложное, то его удобно вводить с помощью мастера

Установка таймера

Оператор стартует таймер, задавая значение интервала времени, после исчерпания которого таймер остановится. Атрибуты оператора:

Имя	Имя таймера. Мастер выбора таймера предлагает полный список таймеров, определённых в процессе
Значение	Число или произвольное арифметическое выражение. Если выражение сложное, то его удобно вводить с помощью мастера
Единица	Значение интервала времени можно задавать в секундах, миллисекундах или микросекундах. Отметим, что не все типы процессоров реально способны задавать интервал с микросекундной точностью

Посылка значения в канал

Оператор начинает передачу значения в канал, принадлежащий какому-либо процессу, причем неважно, будет ли это процесс данного узла или процесс, принадлежащий другому узлу. Если канал принадлежит одному из процессов данного узла, то передача будет выполнена очень быстро, практически за то же время, что и установка значения переменной. Если же канал принадлежит процессу другого узла, то сообщение будет поставлено в очередь передачи к шлюзу (сетевому драйверу).

Очень важно понимать, что передача между узлами не может считаться абсолютно надёжной. Например, связь может быть оборвана, очередь шлюза переполнена, в процессе передачи данные искажены или потеряны из-за помехи и так далее. Если необходимо обеспечить абсолютно надёжную передачу, то потребуется создать обратный канал. Если процесс-получатель получил сообщение по такому каналу, то он должен вернуть его по ответному каналу процессу-отправителю. Если же сообщение было потеряно по любой причине, то процесс-отправитель не получит ответного сообщения. Для обеспечения надёжности процесс-отправитель посылает данные в канал, запускает таймер и ожидает ответной реакции в течении некоторого времени (таймаута). Если ответа не последовало, то процесс-отправитель может сделать ещё 2-3 попытки передачи и в случае неудачи перейти

на альтернативный сценарий, который учитывает невозможность взаимодействия процессов. Конечно, это более сложная последовательность, но она позволяет сохранить работоспособность при любых нарушениях связи в локальной сети или отказе отдельных узлов сети. Пример надежной передачи приведен в проекте «Пример2».

Для уменьшения нагрузки на сеть целесообразно посылать данные по каналам как можно реже – только тогда, когда значение канала действительно должно быть изменено. Быстрая циклическая передача одного и того же значения будет приводить к замедлению работы всех узлов сети и к тому, что действительно полезные передачи могут быть потеряны из-за переполнения буферов сетевых драйверов. Атрибуты оператора:

Имя	Имя канала. Мастер выбора канала предлагает полный список каналов, определённых во всех узлах и во всех процессах проекта
Значение	Число или произвольное арифметическое выражение. Если выражение сложное, то его удобно вводить с помощью мастера.

Посылка значения в устройство

Оператор устанавливает свойство устройства (или активизирует его действие). В любом случае устройство выполняет какое-то действие. Часть устройств, содержащих только одно свойство, допускает не указывать имя свойства (например, устройства-коммутаторы), но большинство устройств требует явного указания и имени устройства и имени свойства. Атрибуты оператора:

Имя	Имя устройства (или имя устройства и его свойства). Мастер выбора устройства предоставляет полный список устройств, определённых в процессе и для каждого устройства даёт список его свойств
Значение	Число или произвольное арифметическое выражение. Если выражение сложное, то его удобно вводить с помощью мастера

Выражения

Выражения встречаются во многих операторах языка Anima. Выражения можно разделить на следующие группы:

1. Условное выражение. Используются в условных операторах и операторе цикла.
2. Арифметическое выражение. Используется в операторе присваивания значения переменной, операторе установки таймера и в операторах передачи данных в канал и в устройство.
3. Константное арифметическое выражение. Используется при задании значения объекта-константы.

Наиболее ограниченным является последний тип выражения. Он позволяет использовать только числа и константы. В константном выражении допустимы только арифметические и побитовые операторы. Арифметическое выражение позволяет использовать константы, переменные, каналы, устройства, таймеры и функции. Условное выражение позволяет дополнительно использовать операторы сравнения и логические операторы.

Все выражения разрешают использовать скобки для группирования подвыражений.

Если выражение сложное, то для его ввода удобно использовать мастер. Мастер позволяет вводить в выражение имена переменных, констант, таблиц, таймеров, каналов, устройств и функций. Для большинства объектов он предоставляет справочную подсказку. Арифметические, логические операторы, скобки и числа вводятся как обычный текст. Далее подробно описываются объекты, используемые выражениях.

Число

Это элементарная константа, например: 1, 23, 1024. Число может быть также задано в шестнадцатиричном и двоичном виде. Шестнадцатиричное число обозначается префиксом 0x, например, 0x3AF2, а двоичное число – префиксом 0b, например, 0b100110. На панели инструментов мастера имеются кнопки для ввода префиксов.

Переменная

Переменная задаётся своим именем.

Константа

Константа задаётся своим именем. Отметим, что в выражениях можно использовать константы, принадлежащие любому узлу сети. Имя константы может состоять из:

- имени константы, если константа принадлежит данному процессу,
- из имени процесса и константы (соединённых точкой), если константа принадлежит другому процессу данного узла,
- из имени узла, имени процесса и имени константы, соединённых точкой, если константа принадлежит процессу другого узла.

Элемент таблицы

Элемент задаётся именем таблицы и индексным выражением в квадратных скобках, например:

```
таблица[4]
```

Индекс может быть числом или произвольным арифметическим выражением. Значение

индекса должно изменяться в пределах от 0 до размера таблицы минус 1. Если индекс меньше 0, то результатом будет первый (нулевой) элемент таблицы. Если индекс больше или равен размеру таблицы, то результатом будет последний элемент таблицы.

Таймер

Таймер задаётся своим именем. Если интервал времени таймера истёк, то значение таймера будет истинным (равным 1), в противном случае значение таймера будет ложным (равным 0). Если таймер ещё не был установлен, то значение таймера неопределено и может быть любым.

Готовность канала

Готовность канала задаётся именем канала и атрибутом «готов», например:

```
канал.готов
```

Имя канала может состоять из:

- имени, если он принадлежит данному процессу,
- из имени процесса и канала (соединённых точкой), если канал принадлежит другому процессу данного узла,
- из имени узла, имени процесса и имени канала, соединённых точкой, если канал принадлежит процессу другого узла.

Если готовность канала равна 0, то канал пуст. Если готовность канала неравна 0, то канал содержит принятое данное. Получать состояние готовности можно только однократно — после получения состояния готовности оно автоматически сбрасывается в 0.

Значение канала

Получать значение канала можно только после проверки его готовности. Значение канала остается неизменным вплоть до следующей передачи в канал. Если значение канала используется несколько раз, то рекомендуется сразу после успешной проверки готовности поместить значение в переменную и использовать её, так как значение канала может быть переписано новым значением. Если значение канала используется только один раз, то его следует получать сразу же после проверки готовности.

Устройство

Имя устройства может состоять из:

- имени устройства, если устройство имеет только одно свойство по умолчанию - <состояние>,
- имени устройства и имени свойства, соединённых точкой.

Все свойства разрешают и чтение и запись данных, но некоторые свойства предназначены только для чтения – запись в них не будет иметь никакого эффекта, а некоторые свойства – только для записи, чтение из них может давать неопределённый или нулевой результат. Это подробно оговаривается для каждого конкретного устройства и свойства.

Функции

Язык Anima определяет несколько функций, которые могут быть использованы в выражениях.

размер(имя_таблицы)

Функция возвращает размер (число элементов) заданной таблицы. Например, если таблица tbl задана таким массивом чисел: 1;9;12;8, то функция

```
размер(tbl)
```

вернет в результате число 4.

случай(мин, макс)

Функция возвращает случайное число в пределах от минимального до максимального включительно. Оба значения могут быть произвольными арифметическими выражениями. Например, функция

```
случай(5, 10)
```

вернет в результате случайным образом одно из следующих чисел: 5,6,7,8,9,10.

максимум(выражение1, выражение2)

Функция возвращает максимальное из двух значений, каждое из которых задано произвольным арифметическим выражением. Например, функция

```
максимум(5, 6)
```

вернет в результате 6.

минимум(выражение1, выражение2)

Подобна предыдущей функции, но возвращает минимальное из двух значений.

rgb(красный, зеленый, синий)

Функция возвращает значение цвета из его компонентов. Например, функция

```
rgb(255, 0, 0)
```

вернет в результате 0xFFFF0000. Старшая компонента цвета это прозрачность, ее значение 0xFF (255) означает полностью непрозрачный цвет.

Арифметические операторы

+	Сложение
-	Оператор может выступать в двух формах, бинарной и унарной. Бинарная операция вычитает второй операнд из первого, а унарная – изменяет знак числа на противоположный
*	Умножение
/	Деление. Результат деления на 0 даёт в результате 0. Чтобы не получать некорректный результат, нужно явно контролировать значение делителя
%	Остаток от деления. Подобно операции деления, остаток от деления на 0 даёт 0

Если операции сложения, вычитания, умножения и деления не требует пояснений, то операция взятия остатка от деления может быть непонятной, а точнее, может быть непонятным её использование. Дадим пару примеров применения этой операции.

1. Допустим у нас есть таймер, отсчитывающий секунды – при каждом срабатывании таймера переменная «время» увеличивается на 1. Нужно получить значение часов, минут и секунд:

```
часы = время / 3600
```

```
минуты = (время % 3600) / 60
секунды = время % 60
```

2. Допустим мы хотим сформировать псевдослучайную последовательность номеров треков аудиопроигрывателя не пользуясь функцией «случай», чтобы исключить многократные последовательные повторения одного и того же трека:

```
таблица 1;3;9;2;6;4;7;5;8;2;3;6;1;8;5;4;6;7; и т.д.
индекс = индекс + 1
трек = таблица[индекс % размер(таблица)]
```

Первая строка этой программы определяет объект-таблицу номеров треков. Если размер таблицы сделать достаточно большим, то выбор треков будет очень похож на случайный. Вторая строка программы увеличивает индекс трека на 1. Третья строка получает номер следующего трека из таблицы. Оператор % используется для того, чтобы привести монотонно возрастающую последовательность индексов к диапазону от 0 до размер таблицы минус 1. Без оператора % пришлось бы делать условный оператор, который проверяет выход значения индекса за пределы размера таблицы и обнуляет индекс.

Побитовые операторы

&	Побитовое «и»
|	Побитовое «или»
~	Побитовая инверсия «не»
^	Побитовое «исключающее или»
<<	Сдвиг влево. Левый операнд сдвигается влево на число бит, равное правому операнду. Правый операнд должен быть в пределах от 0 до 31
>>	Сдвиг вправо. Левый операнд сдвигается вправо на число бит, равное правому операнду. Правый операнд должен быть в пределах от 0 до 31

Битовые операции удобны в том случае, когда нужно упаковать несколько двоичных флагов в одно значение, например, для передачи по каналу. Приведём пример. Допустим, нам нужно передать по каналу данные о состоянии трех кнопок: К1, К2 и К3. Каждая кнопка имеет значение 1, если она нажата и 0 в противном случае. Наиболее эффективная программа, которая формирует общее значение, учитывающее состояние всех кнопок, будет выглять так:

```
значение = К1 | (К2 << 1) | (К3 << 2)
```

Если К2 нажата, то к значению будет логически добавляться 1, сдвинутая влево на 1 разряд (то есть, число 2). Если К3 нажата, то к значению будет добавляться 1, сдвинутая влево на 2 разряда (то есть, число 4). Таким образом, кнопке К1 будет соответствовать нулевой бит в значении, К2 – первый бит и К3 – второй бит. Если кнопка нажата, то соответствующий бит будет установлен в 1, а иначе – в 0.

На другом конце канала значения кнопок можно получить так:

```
К1 = значение & 0b001
К2 = значение & 0b010
К3 = значение & 0b100
```

Если нулевой бит значения равен 1, то К1 будет равно 0b001, если первый бит значения равен 1, то К2 будет равно 0b010, если второй бит значения равен 1, то К3 будет равно 0b100, то есть, если бит равен 1, то значение соответствующей кнопки будет неравно 0, если же бит равен 0, то значение соответствующей кнопки будет равно 0.

Логические операторы

и	Логическое «и». Результат операции истинен (неравен 0), если истинны оба операнда
или	Логическое «или». Результат операции истинен, если истинен хотя бы один операнд
!	Логическое «не». Результат истинен, если операнд ложен и результат ложен, если операнд истинен
=	Проверка равенства. Результат истинен, если левый операнд равен правому
!=	Проверка неравенства. Результат истинен, если левый операнд не равен правому
>=	Проверка «больше или равно». Результат истинен, если левый операнд больше правого операнда или равен ему
>	Проверка «больше». Результат истинен, если левый операнд строго больше правого операнда
<=	Проверка «меньше или равно». Результат истинен, если левый операнд меньше правого операнда или равен ему
<	Проверка «меньше». Результат истинен, если левый операнд строго меньше правого операнда

Приоритеты операторов

Результат сложного выражения, в котором присутствуют различные операторы, зависит от приоритета операций. В арифметике принято что умножение и деление более приоритетны, чем сложение и вычитание. Например, выражение:

```
a + b * c
```

будет вычисляться как:

```
a + (b * c)
```

а не как:

```
(a + b) * c
```

Все операторы разделены на группы с различным приоритетом, чем выше группа в таблице приоритетов, тем больше приоритет её операций. Операторы с одинаковым приоритетом выполняются слева направо:

Приоритет	***Операторы***
0 (высший)	! ~ - (минус унарный)
1	* / % & | ^ << >>
2	+ -
3	= != >= > <= <
4 (низший)	и или

Например, выражение:

```
a + b * c > 3 и d - e * f < -3
```

будет вычисляться как:

```
((a + (b * c)) > 3) и ((d - (e * f)) < (-3))
```

Если возникают сомнения, то в сложном выражении лучше явно расставлять скобки.

Шлюзы и сети

Шлюз RS-232

Этот шлюз позволяет создавать сеть из двух узлов, расположенных на небольшом расстоянии друг от друга. Это очень простая сеть, работающая по стандартному протоколу RS-232. Стандарт RS-232 был разработан Ассоциацией помышленных средств связи TIA и ассоциацией электронной промышленности EIA. Достоинство RS-232 в том, что он поддерживается практически всеми контроллерами и компьютерами. Для персональных компьютеров электронная промышленность выпускает преобразователи USB-Com, которые эмулируют RS-232, а также отдельные карты, содержащие множество каналов RS-232.

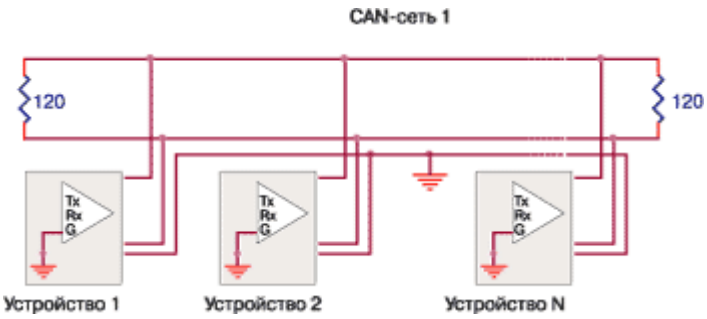
Интерфейс трёхпроводной, содержит линии приёма, передачи и земли. Для увеличения надежности требует специального экранированного кабеля. Максимальная допустимая скорость составляет 115200 бит в секунду, максимальная длина кабеля зависит от скорости и от качества кабеля и составляет от 1 до 15 метров.

Различные процессоры по-разному именуют этот шлюз. Например, в микроконтроллере LPC2368 шлюз RS-232 называется UART0-UART3, а в персональных компьютерах — Com1-Com9.

Шлюз CAN

Это наиболее важный шлюз, который позволяет подключать микроконтроллер к CAN-сети. Такая сеть может иметь множество узлов, причём узлы могут располагаться на довольно большом расстоянии друг от друга. Стандарт CAN был разработан фирмой Bosch и широко применяется в автомобильной и авиационной промышленности, в разнообразных системах автомазации.

Интерфейс трёхпроводной и выполняется с помощью витой пары или специального кабеля:



Максимально допустимая скорость составляет 1 мегабит в секунду. Максимальная длина кабеля зависит от скорости и качества кабеля и находится в пределах от 10 метров до 1 километра. CAN имеет аппаратные средства обнаружения и исправления ошибок приемо-передачи в условиях даже сильных помех. CAN поддерживается многими современными микроконтроллерами, существуют также отдельные микросхемы этого интерфейса, а для персональных компьютеров выпускаются карты-контроллеры CAN.

Важное замечание. Для физической связи следует обязательно использовать витую пару. Экранирование витой пары необязательно. Для связи следует использовать специальные кабели, например, стандартный кабель Ethernet. Ни к коем случае не следует делать связь двумя одножильными экранированными проводами — требуется именно витая пара. По концам кабеля необходимо установить терминаторы — резистры с сопротивлением 120 ом. К одной связи может быть подключено много микроконтроллеров (как показано на рисунке),

но терминаторов должно быть только два — на обоих концах связи. Все сегменты связи должны быть сделаны кабелем одного типа.

Топологии сети

С помощью шлюзов можно строить как однородные сети, использующие только один протокол, так и разнородные сети, использующие несколько протоколов. В последнем случае шлюзы соединяют подсети, работающие по разным протоколам в единую сеть, так что становится неважным, в какой конкретно подсети расположен тот или иной узел. В любом случае взаимодействие параллельных процессов в различных узлах производится с помощью каналов независимо от физического размещения узлов.

Важное требование к топологии сети состоит в том, что сеть может быть только линейной или радиальной, но не может быть кольцевой и не должна содержать замкнутых кольцевых контуров.

Процессоры

Все процессоры разделяются на 2 основных класса — микроконтроллеры и исполняющие системы персонального компьютера.

Процессор «LPC2368»

Микроконтроллер LPC2368 с системой команд ARM7 разработан компании Philips (NXP). Этот 32х разрядный микроконтроллер имеет 512 KB flash-памяти программ, 58 KB памяти данных и может работать на частоте до 72 MHz. В состав периферии микроконтроллера входят следующие устройства:

- Мощный векторный контроллер прерываний;
- Ethernet 10/100 MAC контроллер с DMA, позволяющий работать по протоколу TCP/IP в Internet и Intranet;
- Полноскоростной USB 2.0 контроллер;
- Двухканальный контроллер CAN 2.0B;
- DMA контроллер;
- 4 контроллера UART;
- 3 контроллера последовательного интерфейса I2C;
- 3 контроллера последовательного интерфейса SPI/SSP;
- один контроллер последовательно интерфейса I2S;
- контроллер интерфейса SD/MMC;
- 10-разрядный шестиканальный аналогоцифровой преобразователь;
- 10-разрядный цифроаналоговый преобразователь:
- четыре независимых 32-битных таймера с функциями захвата и сравнения;
- сторожевой таймер;
- широтно-импульсный модулятор для трёхфазного шагового двигателя;
- часы реального времени с автономным батарейным питанием;
- 69 линий ввода-вывода, которые можно использовать для прямого управления различными внешними устройствами.

Электронной промышленностью разработано большое количество драйверов различных устройств – датчиков, преобразователей, драйверов двигатей, которые можно подключать либо к многочисленным последовательным интерфейсам микроконтроллера, либо к линиям ввода-вывода. Встроенный аналогоцифровой преобразователь позволяет строить механизмы с обратной связью по нескольким координатам с точным позиционированием механизма. Встроенные контроллеры Ethernet, CAN, USB, UART позволяют разрабатывать разнообразные шлюзы для распределенной сети, включая связь по Internet.

Для процессора LPC2368 разработаны такие шлюзы:

- UART0 – это специфическое для LPC2368 название шлюза RS-232, в нем используется нулевой канал UART;
- CAN1 – это специфическое название шлюза CAN, в нем используется первый канал CAN.

Шлюзы других типов будут разрабатываться по мере необходимости.

Процессор поддерживает стандарт DMX512. Канал процессора, используемый для DMX512 указывается в атрибутах узла. Возможные каналы:

- TXD2 – используется передатчик UART2.

Устройство «Датчик»

Устройство этого типа принимает логический сигнал с одной из линий ввода-вывода микроконтроллера. Датчики могут быть использованы как сами по себе, так и в составе других, более сложных устройств, например, в качестве датчика концевого или промежуточного положения двигателя. Устройство имеет следующие атрибуты:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение датчика
Вывод	Вывод микроконтроллера, который будет использоваться для приема логического сигнала датчика

Устройство имеет одно свойство (без имени), чтение этого свойства возвращает логическое значение датчика – при активном состоянии возвращается 1, при неактивном – 0. Активное состояние соответствует низкому уровню на выводе микроконтроллера, неактивное — высокому. Имя устройства может быть использовано в выражениях точно также, как имя переменной. Например, если устройство-датчик имеет имя «кнопка», то операция присваивания переменной

```
i = кнопка
```

установит значение переменной i равное 1 при активной (нажатой) кнопке и 0 при ненажатой кнопке. Посылка значения в датчик не будет иметь никакого эффекта.

Устройство «Коммутатор»

Устройства этого типа управляют логическим сигналом по одной из линий ввода-вывода микроконтроллера. Коммутаторы могут быть использованы как сами по себе, так и в составе более сложных устройств. Например, для управления реверсивным двигателем может потребоваться два коммутатора. Устройство имеет следующие атрибуты:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение коммутатора
Вывод	Вывод микроконтроллера, который будет использоваться для выдачи логического сигнала коммутатора

Устройство имеет одно свойство (без имени), посылка значения в это свойство будет устанавливать активное или неактивное значение устройства, подключенного к выводу микроконтроллера. Посылка ненулевого значения устанавливает активное состояние, а посылка 0 – неактивное. Активное состояние соответствует низкому уровню на выводе микроконтроллера, неактивное — высокому. Чтение значения из коммутатора возвращает текущее значение коммутатора.

Устройство «DMX слот»

Устройство этого типа управляет одним слотом выходного интерфейса DMX512.

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение слота

Номер	Номер слота в пакете DMX512. Должен быть в пределах от 1 до 512

Значение слота можно устанавливать и получать. При установке слота значение передается в DMX512, при получении возвращается последнее установленное значение.

Устройство «ДвигательШИМ»

Устройство этого типа управляет реверсивным двигателем постоянного тока с широтно-импульсной модуляцией.

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса
Описание	Необязательный комментарий, характеризующий назначение двигателя
Вывод 1	Вывод микроконтроллера, который будет использоваться для выдачи логического сигнала на первый вход драйвера двигателя
Вывод 2	Вывод микроконтроллера, который будет использоваться для выдачи логического сигнала на второй вход драйвера двигателя
Форсировка	Если форсировка разрешена, то остановка двигателя будет производиться путем подачи двух единиц на выводы драйвера двигателя, если же форсировки нет, то остановка двигателя будет выполняться двумя нулями.
Период	Период широтно-импульсной модуляции в миллисекундах. Допустимые значения — 8, 16, 32 и 64 миллисекунды.
Мин. скорость	Минимальная скорость, при которой двигатель работает. При плавном разгоне двигатель начинает с минимальной скорости и тормозится вплоть до минимальной скорости, после чего выключается.

В ходе выполнения сценария можно управлять следующими свойствами двигателя:

Скорость	Значение скорости в пределах от 0 до 256. При скорости 0 двигатель останавливается, а при скорости 256 работает без широтно-импульсной модуляции на полную мощность. При других значениях скоростей время импульсов широтно-импульсной модуляции определяется как скорость * период_ШИМ / 256 Время паузы импульсов ШИМ равно период_ШИМ - время_импульса
Разгон	Время плавного разгона и торможения двигателя. Задается в миллисекундах и может быть в пределах от 0 до 300 секунд (300000 миллисекунд). Во время разгона скорость плавно увеличивается от минимальной до заданной скорости движения, а во время торможения плавно уменьшается от скорости движения до минимальной скорости и затем двигатель отключается.
Время	Время работы двигателя после разгона и вплоть до торможения. Задается в миллисекундах и может быть в пределах от 0 до 300 секунд. Таким образом, общее время работы двигателя равно время_движения + 2 * время_разгона Направление движения задается знаком времени. При положительном

	времени двигатель вращается в прямом направлении, при отрицательном значении времени — в обратном направлении. Задание этого атрибута приводит к старту двигателя с учетом предварительно установленных значений скорости и времени разгона. Если двигатель уже работает, то задание ненулевого времени приводит началу нового движения.
Движение	Старт непрерывного движения со скоростью, зависящей от значения атрибута «Движение». Направление движения задается знаком значения. При положительном значении двигатель вращается в прямом направлении, при отрицательном значении — в обратном направлении, при нулевом значении — останавливается. Значение атрибута определяет скорость движения и может быть задано в пределах от -256 до +256. Атрибут «Движение» не изменяет значение атрибута «Скорость». Атрибуты «Движение» и «Время» взаимоисключающие. Если двигатель уже находился в движении, то просто меняется скорость его движения (и/или направление). Если двигатель был запущен на движение по времени, то временной режим заменяется режимом непрерывного движения. И наоборот, если двигатель был запущен с режиме непрерывного движения и задается атрибут «Время», то режим двигателя заменяется на режим движения по времени (с разгоном и торможением). Атрибут «Движение» может быть использован для управления движением двигателя под управлением сценария
Стоп	Установка атрибута в любое значение останавливает двигатель. Получение значения атрибута можно использовать для контроля факта работы двигателя — единичное значение соответствует остановленному двигателю, а нулевое — работающему двигателю

Кроме своего основного назначения, двигатель с ШИМ используется как основа для других двигателей. Демонстрация использования двигателя с ШИМ дается в примере 10.

Устройство «Двигатель ШИМ, ПИД и импульсным энкодером»

Этот тип устройства добавляет к базовому типу «Двигатель ШИМ» пропорционально-интегрально-дифференциальный (ПИД) регулятор позиции, импульсный энкодер и атрибут задания позиции перемещения. Все свойства базового типа также могут быть использованы. Далее будут указаны только те атрибуты, которые добавляются к атрибутам базового типа. Они делятся на две группы — атрибуты ПИД-регулятора и атрибуты энкодера.

К_пр	Коэффициент усиления пропорциональной составляющей регулятора
К_инт	Коэффициент усиления интегральной составляющей регулятора
Макс_инт	Максимальное значение интегратора
К_диф	Коэффициент усиления дифференциальной составляющей регулятора
Зона	Разность между текущей позицией двигателя и значением энкодера, при котором начинается ПИД-регулирование. Если разность позиций меньше зоны, то двигатель разгоняется обычным образом с учетом атрибутов Разгон и Скорость до вхождения в зону, а затем начинается процесс регулирования, приводящий к установлению заданной позиции
Таймаут	Максимальное время (в секундах), требуемое для самого длинного

	перемещения. Этот атрибут имеет защитное значение, предотвращающее бесконечное движение двигателя при каких-то неполадках или при неустойчивости процесса ПИД-регулирования

Атрибуты энкодера:

Вывод энкодера 1	Вывод микроконтроллера, который будет использоваться для приема логического сигнала с первого выхода энкодера
Вывод энкодера 2	Вывод микроконтроллера, который будет использоваться для приема логического сигнала с второго выхода энкодера

Назначение выводов двигателя и энкодера должно быть согласовано, то есть, при движении вперед значение энкодера должно увеличиваться, а при движении назад — уменьшаться. В ходе выполнения сценария доступны все атрибуты ПИД-регулятора и следующие атрибуты:

Энкодер	Текущая позиция энкодера, равная числу полученных импульсов. При движении в прямом направлении позиция увеличивается, а при движении в обратном - уменьшается. Позицию можно получать и устанавливать. Установка позволяет задать начальное значение позиции энкодера. Позиция энкодера отслеживается независимо от того, стартован двигатель или нет. Поэтому возможно внешнее (ручное) изменение позиции энкодера без запуска двигателя.
Позиция	Старт движения к указанной позиции. Время движения ограничено таймаутом. Максимальная скорость движения задается атрибутом «Скорость». Алгоритм движения зависит от разности текущей позиции двигателя и заданной позиции. Если разность велика (превышает зону), то двигатель сначала разгоняется. ПИД-регулирование начинается только при входжении разности в зону регулирования

Демонстрация использования двигателя дается в примере 11. С помощью этого примера также можно подобрать настройки ПИД-регулятора и другие атрибуты двигателя.

Настройка ПИД-регулятора зависит от множества факторов — типа двигателя, особенностей механизма, частоты импульсов энкодера, требований к движению. Начинать настройку нужно с установки коэффициентов усиления интегральной и дифференциальной составляющих в 0. Затем подбирается коэффициент усиления пропорциональной составляющей. Его среднее значение зависит от частоты импульсов энкодера и может быть установлено в 1000. Увеличение значения приводит к увеличению точности регулирования, но также и к колебательному характеру подхода к заданной позиции. Уменьшение значения снижает точность регулирования, по дает плавное приближение к заданной позиции. Если качество приближения к позиции не устраивает, по вводится интегральная часть — обычно она составляет несколько процентов от пропорциональной части. Интегральная часть обеспечивает точность регулирования но приводит у увеличению времени регулирования. Максимальное значение интегратора определяет максимальную поправку к интегральной части. Начинать ее подбор можно с установки, равной значению пропорциональной части. Чем больше максимальное значение, тем точнее будет регулирование, но время регулирования также будет увеличиваться. Дифференциальная составляющая позволяет увеличить скорость регулирования при коротких перемещениях.

Устройство «Двигатель ШИМ, ПИД и абсолютным энкодером EAWOJB24AEO128L»

Этот тип устройства подобен предыдущему, и отличается наличием 8 выводов для абсолютного энкодера (а не 2 для импульсного энкодера). Все остальные атрибуты имеют аналогичное значение.

Настройка ПИД-регулятора имеет некоторую особенность по сравнению с регулятором для двигателя с импульсным энкодером. Она состоит в том, что число позиций абсолютного энкодера намного меньше, чем у импульсного энкодера, поэтому все коэффициенты усиления обычно увеличены во много раз. Например, если коэффициент усиления пропорциональной составляющей для двигателя с импульсным энкодером в среднем составляет 1000, то для двигателя с абсолютным энкодером — 100000.

Демонстрация использования двигателя дается в примере 12. С помощью этого примера также можно подобрать настройки ПИД-регулятора и другие атрибуты двигателя.

Устройство «Абсолютный энкодер EAWOJB24AEO128L»

Этот тип устройства используется для создания джойстиков-манипуляторов, используемых для интерактивного управления аппаратурой. Логически устройство подобно обычному датчику, но в отличие от него возвращает значение в диапазоне от 0 до 127, соответствующее позиции энкодера. Устройство имеет атрибуты назначения для восьми выводов микроконтроллера. По ходу сценария позицию энкодера можно получать с помощью атрибута «Позиция».

Демонстрация использования двигателя дается в примере 13. С помощью этого примера также можно проверить правильность назначения выводов микроконтроллера.

Процессор «LPC2378»

Возможности этого процессора практически эквивалентны возможностям процессора LPC2368, основное отличие их состоит в количестве доступных выводов.

Процессор «WF»

Типом WF обозначается персональный компьютер, который работает под управлением операционной системы Windows с исполняющей системой Windows Forms (WF). Эта графическая исполняющая система позволяет строить пульты управления, включающие кнопки, регуляторы, индикаторы. В этой исполняющей системе отсутствуют такие мультимедийные возможности, как видео-аудио-проигрыватели.

Для процессора WF разработаты шлюзы:

- Com (RS-232), в зависимости от номера порта шлюз называется Com1–Com9

Устройство «Окно»

Это устройство специального назначения, которое управляет другими экранными устройствами. Экранные устройства обладают атрибутами двух типов — статического и динамического. Статические атрибуты задаются инспектором объектов, но не могут быть изменены в ходе работы сценария. Динамические атрибуты, напротив, задаются только в сценарии. Существуют также смешанные атрибуты, которые одновременно являются и статическими и динамическими, то есть, их начальное значение задается инспектором объектов, но может быть также изменено в сценарии.

Окно может принимать коды нажимаемых клавиш и передавать их в канал. Код клавиши состоит из собственно кода и дополнительного модификатора, зависящего от клавиш Shift, Ctrl, Alt. Структура полного кода клавиши следующая:

- Биты 0..7 — собственно код клавиши
- Бит 8 — признак нажатой клавиши Shift,
- Бит 9 — признак нажатой клавиши Ctrl,
- Бит 10 — признак нажатой клавиши Alt.

Обычная процедура работы с кодом клавиш может быть выражена так:

```
клавиша = код & 0b00011111111 // 0x0FF
shift   = код & 0b00100000000 // 0x100
ctrl    = код & 0b01000000000 // 0x200
alt     = код & 0b10000000000 // 0x400
```

Пример работы с клавишами дан в проекте «Пример4». В этом же примере определена группа констант — кодов клавиш. Эту группу можно копировать в другие проекты, чтобы получить соответствие кодов клавиш и их названия и использовать названия клавиш вместо цифровых кодов.

Отметим, что некоторые сочетания клавиш имеют специальное назначение:

- Shift+Esc — закрытие окна и выход из программы,
- F1 – вызов инспектора. Эта клавиша работает только в интегрированной среде проектирования KoskaXmlEditor и отображает инспектор объектов,
- Shift+F1 – вызов отладчика,
- Tab – переход между устройствами в окне,
- Пробел, Enter, Home, End, Up, Down, Left, Right, PageUp, PageDown – эти клавиши имеют определенное значение для некоторых устройств, которые находятся в фокусе.

Атрибуты окна:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение	статический

	устройства	
Имя канала	Этот атрибут задает имя канала, в который окно будет посылать коды нажимаемых клавиш. Если управление от клавиатуры не требуется, то имя канала можно не задавать	статический
X	Координата левой стороны окна на экране. С помощью координаты X можно указать первичный или вторичный монитор. Например, если первичный монитор имеет ширину (разрешение) 1024 пикселя, то указание X=1024 разместит окно по левому краю вторичного монитора	статический
Y	Координата верхней стороны окна на экране	статический
Ширина	Ширина окна в пикселях. Если атрибут равен 0, то ширина окна автоматически устанавливается равной ширине первичного монитора	статический
Высота	Атрибут подобен предыдущему, но задаёт высоту окна	статический
Режим	Окно может быть обычным или полноэкранным — в этом случае размер окна устанавливается равным размеру экрана. В полноэкранном режиме окно не имеет заголовка и рамки	статический
Заголовок	Текст в области заголовка окна. Имеет смысл только в оконном режиме	статический
Цвет фона	Цвет рабочей поверхности окна	статический
Цвет текста	Цвет текста по умолчанию, который используется для всех вложенных устройств, если в них цвет текста явно не переопределен	статический
Шрифт	Шрифт текста по умолчанию, который используется для всех вложенных устройств, если в них цвет текста явно не переопределен	статический

Устройство-окно может содержать только устройства типа «Страница». Редактирование вложенных устройств окна выполняется обычным образом с помощью инспектора, но удобнее редактировать атрибуты устройств в режиме предварительного просмотра. Для этого нужно выбрать окно (или его вложенное устройство) и выполнить запрос «Предварительный просмотр окна». В этом режиме окно будет отображаться точно также, как и при выполнении готовой программы, но сценарий в этом режиме не выполняется и окно не реагирует на мышку и клавиатуру. Для редактирования атрибутов устройств окна нужно нажать в предварительном просмотре кнопку F1 – при этом на экране будет отображена копия инспектора и дерево объектов окна.

В этом инспекторе свойств нельзя создавать новые устройства, но можно изменять их атрибуты и сразу же наблюдать результаты произведенных изменений. Для выхода из режима предварительного просмотра нужно закрыть инспектор объектов и окно.

Пример использования всех устройств процессора WF дан в проекте «Пример5».

Устройство «Страница»

Страницы позволяют сгруппировать устройства таким образом, чтобы видеть в окне только часть устройств и не загромождать окно излишними устройствами. Для перехода между страницами сценарий должен делать текущую страницу невидимой, а новую страницу — видимой. Пример работы со страницами дан в проекте «Пример5». Атрибуты страницы:

Имя устройства	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Видимость	Определяет, видима ли страница. Этот атрибут является смешанным, то есть, может быть установлен как в инспекторе объектов, так и в ходе работы сценария. В инспекторе нужно установить видимость только одной, начальной, страницы. Все остальные страницы должны быть невидимыми	смешанный

Страница может содержать следующие вложенные устройства:

- Панель
- Текст
- Регулятор
- Кнопка
- Счетчик
- Флажок
- Рисунок

Устройство «Панель»

Устройство служит для группировки других устройств и цветового выделения группы. С помощью Панели можно отображать не только сплошной фон, но и рамку — для этого нужно вложить одну панель в другую и немного сдвинуть вложенную панель и изменить ее размеры. В зависимости от сдвига панелей будет изменяться толщина рамки. Следует отметить, что координаты X и Y вложенных устройств отсчитываются от 0, то есть, X=1 означает смещение вложенного устройства по отношению к родительскому устройству на 1 пиксель вправо. Панель может содержать в себе те же устройства, что и страница. Атрибуты панели:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
X	Горизонтальная координата левой стороны панели относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны панели относительно родительского устройства	статический
Ширина	Ширина панели	статический
Высота	Высота панели	статический
Видимость	Видима ли панель?	смешанный
Цвет фона	Цвет, которым будет закрашиваться поверхность панели. Мастер выбора цвета позволяет установить цвет на основе трех цветовых составляющих RGB, выбрать один из стандартных Web-цветов или один из стандартных Windows-цветов. Отметим, что цвет фона является смешанным атрибутом, то есть, он может быть установлен как в инспекторе, так и по ходу сценария. Цвет задается в формате	смешанный

	RGB, получить значение цвета из отдельных компонентов можно с помощью функции rgb().	

Устройство «Текст»

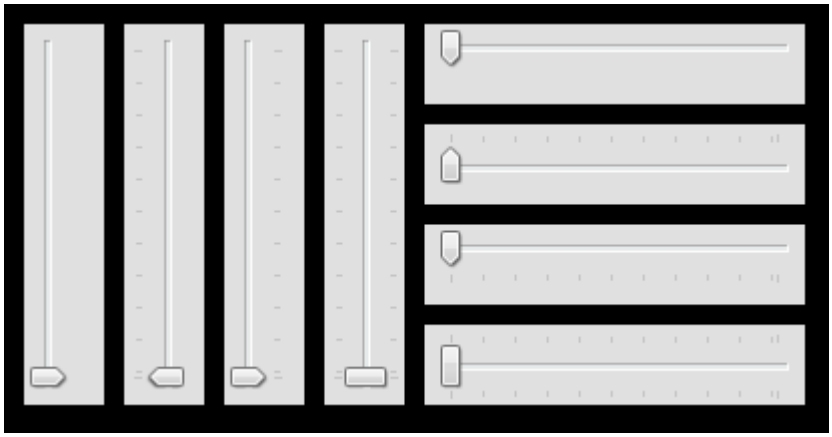
Устройство отображает текст или числовое значение. Текст не может содержать вложенных устройств. Атрибуты текста:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
X	Горизонтальная координата левой стороны текста относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны текста относительно родительского устройства	статический
Ширина	Ширина текстового прямоугольника	статический
Высота	Высота текстового прямоугольника	статический
Видимость	Видим ли текст?	смешанный
Текст	Текст, отображаемый устройством	статический
Выравнивание	Способ выравнивания текста внутри текстового прямоугольника. Возможные варианты выравнивания: влево, вправо, по центру	статический
Цвет фона	Цвет, которым будет закрашиваться текстовый прямоугольник. Если задать прозрачный цвет (Transparent) или вообще его не задавать, то текстовый прямоугольник будет невидимым	смешанный
Цвет текста	Цвет, которым будет отображаться текст. Если цвет не задан, то используется цвет родительского устройства	смешанный
Шрифт	Шрифт, которым будет отображаться текст. Если шрифт не задан, то используется шрифт родительского устройства	статический
Значение	Это динамический атрибут, его можно установить только по ходу сценария. Установка этого атрибута отображает числовое значение вместо статического атрибута «Текст»	динамический

Устройство «Регулятор»

Устройство отображает ползунковый регулятор, положение которого можно менять с помощью мышки. Если регулятор выбран (находится в фокусе), то его состояние можно менять также и с помощью клавиш Home (в конечное значение), End (в начальное значение), Вверх (увеличение маленькими шагами), Вниз (уменьшение маленькими шагами), PageUp (увеличение большими шагами), PageDown (уменьшение большими шагами).

Возможные варианты отображения регулятора:



Атрибуты регулятора:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который регулятор будет посылать свое значение при перемещении ползунка. Этот атрибут можно не указывать, если сценарий будет просто запрашивать значение регулятора по ходу сценария	статический
X	Горизонтальная координата левой стороны регулятора относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны регулятора относительно родительского устройства	статический
Ширина	Ширина прямоугольника регулятора	статический
Высота	Высота прямоугольника регулятора	статический
Видимость	Видим ли регулятор?	смешанный
Максимум	Максимальное значение, которое может принимать регулятор. Значение регулятора меняется в пределах от 0 до максимума	статический
Значение	Текущее значение регулятора — его можно устанавливать и получать. Попытка установки значения, большего чем максимум, установит максимальное значение	смешанный
Ориентация	Способ ориентации регулятора. Возможные варианты ориентации: горизонтально или вертикально	статический
Число рисок	Количество рисок, отображаемых около ползунка. Это значение задается без учета начальной (нулевой) риски	статический
Размещение рисок	Способ размещения рисок. Возможные варианты: не отображать риски, размещать слева (или сверху), справа (или снизу), с обоих сторон	статический
Цвет фона	Цвет, которым будет закрашиваться прямоугольник регулятора. Если задать прозрачный цвет (Transparent) или вообще его не задавать, то будет отображаться только	смешанный

	сам ползунок и риски, а прямоугольник регулятора будет невидимым	

Устройство «Кнопка»

Кнопку можно нажать с помощью мышки. Если кнопка находится в фокусе, то ее можно нажать клавишей Пробел. Атрибуты кнопки:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который кнопка будет посылать свое значение при нажатии	статический
Значение	Значение, которое будет посылаться в канал при нажатии кнопки	статический
X	Горизонтальная координата левой стороны кнопки относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны кнопки относительно родительского устройства	статический
Ширина	Ширина кнопки	статический
Высота	Высота кнопки	статический
Видимость	Видима ли кнопка?	смешанный
Текст	Текст, отображаемый на кнопке	статический
Шрифт	Шрифт, которым будет отображаться текст. Если шрифт не задан, то используется шрифт родительского устройства	статический

Устройство «Счетчик»

Устройство позволяет либо явно задать значение, либо ступенчато его изменять. Выглядит счетчик так:

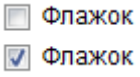
Счетчик имеет текстовое поле, в которое можно ввести значение и затем нажать клавишу Enter. Кроме того, счетчик имеет две кнопки увеличения и уменьшения значений на единицу. Если счетчик находится в фокусе, то изменять значение можно с помощью клавиш Вверх и Вниз. Атрибуты счетчика:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который счетчик будет посылать свое значение при его изменении. Если предполагается считывание значения непосредственно по ходу сценария, то имя канала можно не задавать	статический

X	Горизонтальная координата левой стороны счетчика относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны счетчика относительно родительского устройства	статический
Ширина	Ширина счетчика	статический
Высота	Высота счетчика	статический
Видимость	Видим ли счетчик?	смешанный
Максимум	Максимальное значение, которое может принимать счетчик. Значение регулятора меняется в пределах от минимума до максимума	статический
Минимум	Минимальное значение, которое может принимать счетчик. Минимум может быть отрицательным	статический
Значение	Значение счетчика. Значение можно устанавливать и получать	смешанный
Выравнивание	Способ выравнивания текста внутри текстового поля счетчика. Возможные варианты выравнивания: влево, вправо, по центру	статический
Цвет фона	Цвет, которым будет закрашиваться текстовое поле счетчика. Если цвет не задан, то будет использоваться цвет фона родительского устройства	статический
Цвет текста	Цвет, которым будет отображаться текст значения. Если цвет не задан, то используется цвет родительского устройства	статический
Шрифт	Шрифт, которым будет отображаться текст. Если шрифт не задан, то используется шрифт родительского устройства	статический

Устройство «Флажок»

Устройство этого типа напоминает кнопку с фиксацией, то есть, флажок может быть либо установлен, либо сброшен. Управлять состоянием флажка можно мышкой или клавишей Пробел (если флажок находится в фокусе). Флажок выглядит так:



Атрибуты флажка:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который флажок будет посылать свое значение при его изменении. Если предполагается считывание значения непосредственно по ходу сценария, то имя канала можно не задавать	статический
X	Горизонтальная координата левой стороны флажка относительно родительского устройства	статический

Y	Вертикальная координата верхней стороны флажка относительно родительского устройства	статический
Ширина	Ширина прямоугольника флажка	статический
Высота	Высота прямоугольника флажка	статический
Видимость	Видим ли флажок?	смешанный
Значение	Значение флажка. Значение можно устанавливать и получать. Отмеченному флажку соответствует значение 1, неотмеченному — 0	смешанный
Текст	Текст, отображаемый на флажке	статический
Цвет фона	Цвет, которым будет закрашиваться текстовое поле счетчика. Если цвет не задан, то будет использоваться цвет фона родительского устройства	статический
Цвет текста	Цвет, которым будет отображаться текст значения. Если цвет не задан, то используется цвет родительского устройства	статический
Шрифт	Шрифт, которым будет отображаться текст. Если шрифт не задан, то используется шрифт родительского устройства	статический

Устройство «Рисунок»

Устройство может быть использовано либо как элемент оформления окна, либо как графическая кнопка. В отличие от обычной кнопки, рисунок не может быть в фокусе, и, поэтому, графическую кнопку можно нажать только с помощью мышки. Атрибуты рисунка:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который графическая кнопка будет посылать свое значение при нажатии. Если рисунок не используется как кнопка, то указывать канал не нужно	статический
Значение	Значение, которое будет посылаться в канал при нажатии кнопки	статический
X	Горизонтальная координата левой стороны рисунка относительно родительского устройства	статический
Y	Вертикальная координата верхней стороны рисунка относительно родительского устройства	статический
Ширина	Ширина рисунка. Рисунок растягивается или сжимается в соответствие с указанной шириной и высотой	статический
Высота	Высота рисунка	статический
Видимость	Видим ли рисунок?	смешанный
Файл	Имя графического файла рисунка. Файл должен располагаться в том же каталоге, что и программа	статический

Процессор «WPF»

Типом WPF обозначается персональный компьютер, который работает под управлением операционной системы Windows с исполняющей системой Windows Presentation Foundation (WPF). Эта исполняющая система позволяет строить графические представления с мультимедийными возможностями.

Процессор WPF имеет те же шлюзы, что и WF.

Для указания имен файлов используется файл списка строк.

Устройство «Окно»

Это устройство специального назначения, которое управляет другими экранными устройствами. Окно WPF имеет сходное назначение с окном WF и обладает точно такими же свойства и такими же клавишами специального назначения.

Устройство «Видео»

Устройство предназначено для проигрывания видеофайлов. Устройство отображается на всей поверхности родительского окна. Атрибуты видео:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который устройство будет посылать сообщения о временных событиях. Если отслеживание событий не требуется, то указывать канал не нужно. При наступлении временной метки в канал посылается порядковый индекс метки (так, как он обозначается в программе разметки). В момент начала воспроизведения файла в канал посылается значение 0, а при завершении воспроизведения — отрицательное значение	статический
Видимость	Видимо ли устройство?	смешанный
Файл	Индекс видеофайла в списке строк. Посылка значения в этот атрибут открывает файл и начинает его воспроизведение	динамический
Воспроизведение	Для приостановки проигрывания нужно послать в этот атрибут значение 0, а для продолжения — значение 1. Получение этого атрибута также возможно — результат будет ненулевым, если файл воспроизводится и нулевым, если он приостановлен или завершился	динамический
Громкость	Уровень громкости в пределах от 0 до 1000	смешанный
Баланс	Баланс уровня громкости в стереоканалах. Задается в пределах от -1000 (левый канал) до 1000 (правый канал)	смешанный

Устройство «Аудио»

Устройство предназначено для проигрывания аудиофайлов. Устройство невидимо. Атрибуты аудио-устройства очень похожи на атрибуты видео-устройства:

Имя	Имя устройства должно быть идентификатором и должно быть уникальным для процесса	статический
Описание	Необязательный комментарий, характеризующий назначение устройства	статический
Имя канала	Этот атрибут задает имя канала, в который устройство будет посылать сообщения о временных событиях. События аудио-устройства такие же, как у видео-устройства	статический
Файл	Индекс видеофайла в списке строк. Посылка значения в этот атрибут открывает файл и начинает его воспроизведение	динамический
Воспроизведение	Для приостановки проигрывания нужно послать в этот атрибут значение 0, а для продолжения — значение 1	динамический
Громкость	Уровень громкости	смешанный
Баланс	Баланс уровня громкости в стереоканалах	смешанный

Интегрированная среда KoskaXmlEditor

Все действия по созданию, редактированию и отладке программы проходят в программе KoskaXmlEditor или KoskaXmlEditor2010. Эти программы имеют сходные возможности, являются универсальными и могут поддерживать множество различных типов проектов. В нашем случае используется только один тип – «Проект Anima». Основное различие между двумя программами состоит в том, что KoskaXmlEditor2010 является более новой версией, которая позволяет открывать одновременно много документов, в отличие от KoskaXmlEditor, в которой можно открыть только один документ.

Поскольку краткое описание этой программы было дано в одном из первых разделов данного описания, здесь мы ограничимся только перечислением команд редактора.

Файловые команды

Новый...

Команда создаёт новый проект. При выполнении этой команды отображается список, содержащий только один тип проекта – «Проект Anima». После выбора типа проекта будет создан пустой проект. После того, как проект создан, желательно сразу же сохранить его с каким-то именем. Для каждого проекта должен быть создан отдельный каталог. В этот каталог записывается не только файл проекта,– в нём также будут формироваться результаты компиляции (программы узлов), а также храниться мультимедийные файлы, используемые в проекте.

Открыть...

Команда открывает один из ранее созданных проектов, предлагая выбор каталога и файла проекта.

Закрыть

Команда закрывает текущий проект. Если текущий проект был изменён, но не сохранён, то предлагается выбор – либо сохранить проект, либо выйти без сохранения, либо отменить команду закрытия.

Сохранить

Команда сохраняет изменённый проект на диске. Если проект не был изменён, то команда недоступна.

Сохранить как...

Команда сохраняет проект с новым именем. Можно использовать эту команду сразу после создания нового проекта или для создания новой версии проекта.

Свойства корневого объекта

Команда делает активным корневой (невидимый) объект проекта, что позволяет посмотреть и изменить его атрибуты.

Команды правки (редактирования)

Отменить

Команда отменяет предыдущую команду, связанную с изменением проекта. Например, если предыдущей командой было удаление объекта, то отмена восстановит удалённый объект. Если было изменено свойство объекта, то отмена восстановит предыдущее значение. Количество откатов не ограничено. После записи проекта очередь откатов очищается, то есть после записи проекта на диск отмена действий, предшествующих записи, уже невозможна.

Повторить

Команда отменяет предыдущую команду отмены.

Вырезать

Команды копирует текущий объект со всеми вложенными объектами в буфер обмена и удаляет его из проекта. Эту команду можно использовать для переноса объекта в другую точку проекта, например, в другой процесс или в другой сценарий. Перемещать объекты можно также, буксируя их мышкой. Если необходимо вставить буксируемый объект как вложенный, то при буксировке нужно нажать и удерживать клавишу Shift.

Копировать

Команда копирует текущий объект со всеми вложенными объектами в буфер обмена. Копирование можно использовать для переноса фрагментов из одного проекта в другой. Для этого нужно открыть одновременно два экземляра программы KoskaXmlEditor.

Вставить

Команда вставляет содержимое буфера обмена сразу после текущего оператора. Если в буфер обмена ещё не был скопирован ни один объект, то команда будет недоступной.

Вставить вложенный

Команда подобна предыдущей, но вставляет содержимое буфера обмена как вложенный оператор для текущего оператора.

Удалить

Команда удаляет текущий оператор со всеми сложенными операторами.

Команды управления видом

Обновить

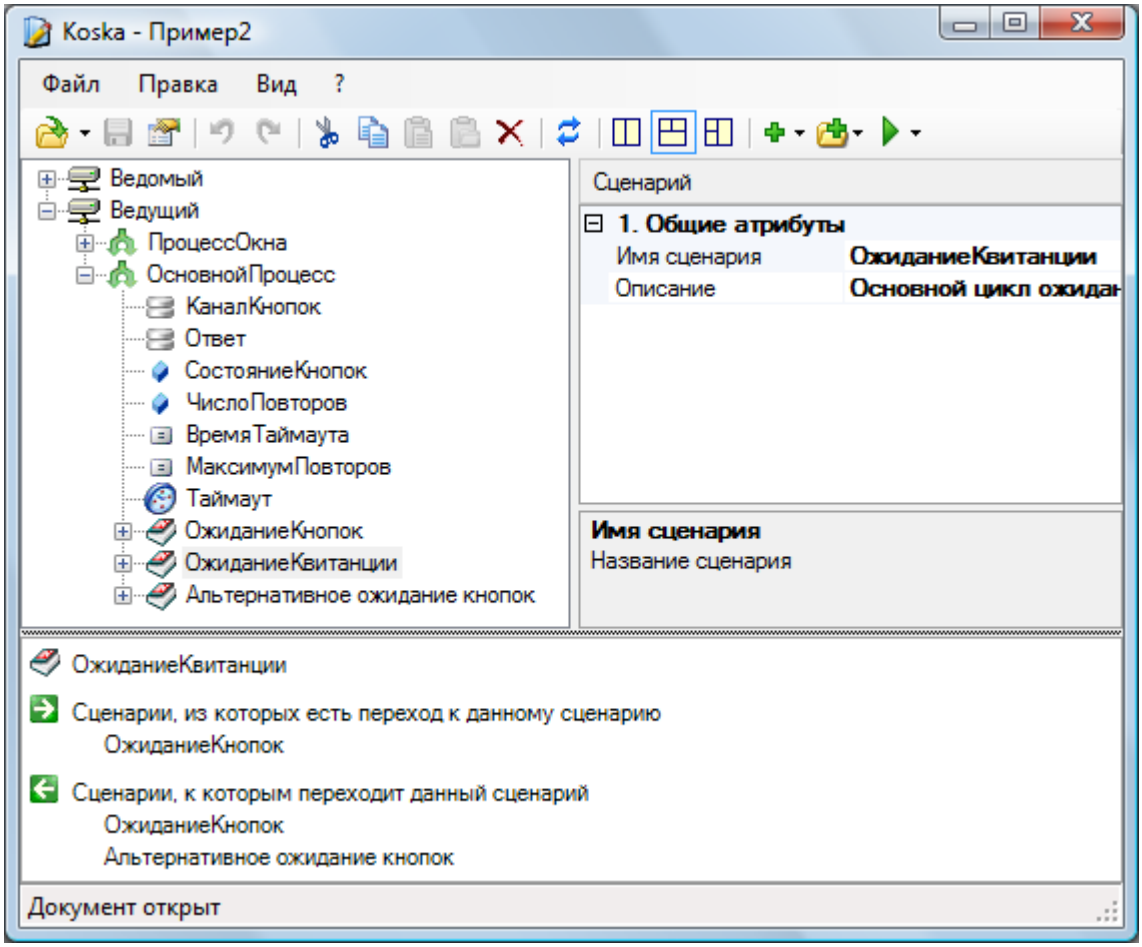
Команда просто обновляет окно редактора. Обычно это не имеет никакого эффекта.

Без просмотра

Команда закрывает область предварительного просмотра. При этом рабочая область программы содержить только древовидное представление объектов проекта и инспектор атрибутов. По умолчанию область предварительного просмотра закрыта.

Горизонтальный просмотр

Команда открывает область предварительного просмотра, разделяя окно проекта и отображая небольшую горизонтально расположенную часть окна, в которой отображается дополнительная графическая информация о текущем операторе проекта. Отметим, что только очень небольшое число операторов могут отображаться в области просмотра. Один из таких объектов – сценарий:



Область предварительного просмотра показывает взаимосвязь текущего сценария с другими сценариями.

Вертикальный просмотр

Команда аналогична предыдущей, но разделяет окно по вертикали.

Команды проекта

Добавить

Команда добавляет новый оператор за текущим оператором. Команды имеет выпадающее меню, которое зависит от того, какой оператор является текущим.

Добавить вложенный

Команда добавляет новый оператор как вложенный для текущего оператора. Если оператор не может иметь вложенных объектов-операторов, то команда будет недоступной. Выпадающее меню команды зависит от текущего оператора.

Выполнить запрос

Команда выполняет запрос к проекту:

Проверить проект. Этот запрос проверяет правильность всех операторов проекта. При ошибке выводится поясняющее сообщение и текущим становится тот оператор, в котором ошибка была обнаружена.

Проверить и скомпилировать проект. Этот запрос не только проверяет проект, но также и компилирует его, то есть, для каждого узла создаёт специальный файл, содержащий программу работы узла. Скомпилированная программа может быть зашита во flash-память микроконтроллера или выполнена на персональном компьютере. Скомпилированный файл имеет имя узла и расширение bin. Кроме файла программы запрос создаёт для каждого узла ещё один файл, используемый при отладке программы. Этот файл имеет имя узла и расширение dbg.

Показать скомпилированный код узла. Это служебный запрос, используется только в том случае, когда есть сомнение в правильности работы компилятора. Запрос помогает обнаружить ошибки при разработке новых устройств. Перед выполнением этого запроса нужно сначала выбрать узел.

Программировать микроконтроллер или запустить программу узла. Запрос вызывает мастер, который производит зашивку скомпилированной программы во flash-память микроконтроллера. Для персонального компьютера этот запрос запускает исполняющую систему для данного типа процессора и передаёт ей файл программы. Перед выполнением запроса нужно сначала выбрать узел.

Показать окно отладчика. Это отладочный запрос, который помогает обнаружить ошибки выполнения параллельных процессов. Окно показывает все переменные, определённые во всех процессах узла. Наблюдая значения переменных можно судить о выполнении программы, для отладочных целей можно даже вводить в сценарии дополнительные переменные, которые показывают, например, какой из сценариев выполняется. Обычно отладка (трассировка) помогает в том случае, если какой-либо сценарий зацикливается в ожидании некоторого события, которое не происходит, причём непонятно, какого конкретно события ожидает сценарий. Запрос может быть выполнен для микроконтроллера, у которого имеется свободный отладочный порт. Кроме того, окно отладки для любого из узлов сети можно вызвать из программы, выполняющейся на персональном компьютере. Например, для процессора LPC2368 и программирование и отладка выполняется через порт UART0. Если же UART0 используется в качестве шлюза, то вызвать окно трассировки можно из любого другого узла сети.

Предварительный просмотр окна. Запрос показывает на экране окно предварительного просмотра, в котором отображаются только видимые устройства. Перед выполнением этого запроса нужно выбрать устройство-окно. Во время предварительного просмотра можно вызвать инспектор объектов (клавишей F1) и подредактировать атрибуты устройств.

Редактировать список строк. Запрос отображает на экране редактор списка строк, используемый в узле. Список строк используется некоторыми процессорами (исполняющими системами компьютера) для хранения имен файлов и другого текста. Список ставит в соответствие целочисленный индекс и текстовое значение. Строки списка имеют формат:

```
индекс=значение
```

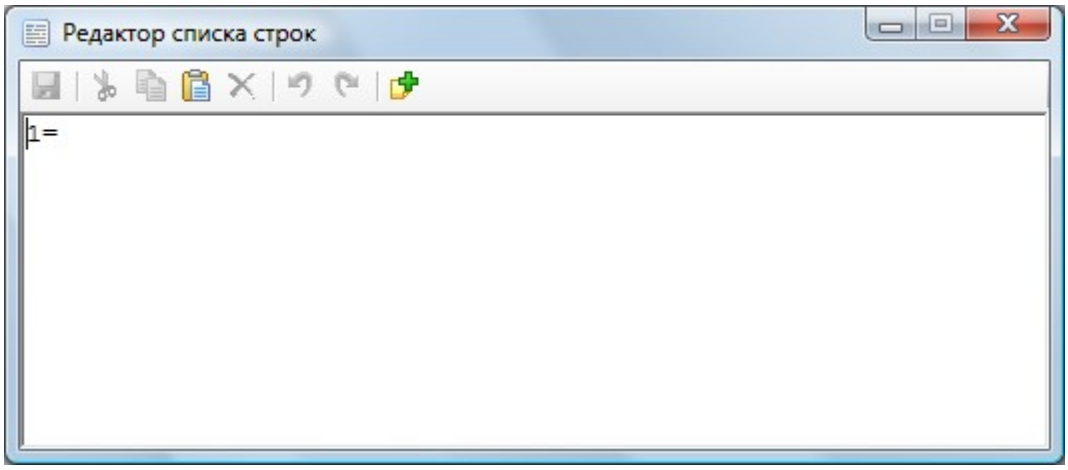
например:

```
43=Небо.avi
```

В этом примере индексу 43 ставится в соответствие имя avi-файла. Поскольку путь файла не указан явно, предполагается, что файл находится в том же каталоге, что и программа узла. Использование индексов позволяет более удобно работать с именами файлов. Например, для выбора случайного звукового файла можно создать список и выбирать файл по случайному индексу. Другое достоинство списка состоит в том, что имена файлов не привязаны к программе узла и могут быть расположены где угодно. При изменении размещения файла достаточно изменить текстовое значение в списке строк, а программа узла останется при этом неизменной. Индексы необязательно делать последовательно возрастающими, главное,

чтобы в списке не было двух строк с одинаковыми индексами.

Окно редактора списка:

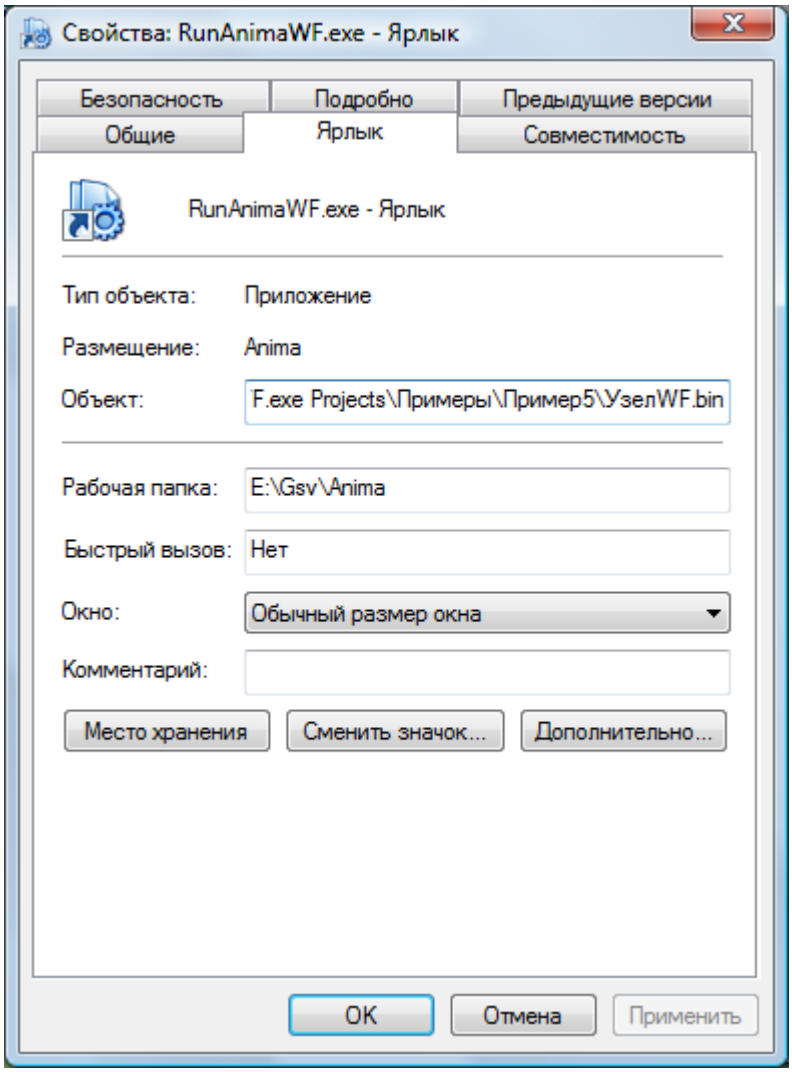


Редактор имеет обычные команды текстового редактирования и одну дополнительную — вставка новой строки. Для вызова редактора должен быть выделен узел. Имя файла списка совпадается с именем узла и имеет расширение lst.

Исполняющая система RunAnima

Для запуска программы на микроконтроллере нужно предварительно зашить в него исполняющую систему и набор драйверов, которые специфичны для микроконтроллера каждого конкретного типа. Для персональных компьютеров под управлением Windows исполняющая система представлена набором программ RunAnimaX.exe, где X означает тип процессора, в частности, RunAnimaWF.exe и RunAnimaWPF.exe. Этой программе нужно передать имя файла, которое содержит скомпилированный код для узла. При отладке её можно запускать из интегрированной среды KoskaXmlEditor с помощью запроса «Программировать микроконтроллер и запустить программу узла». Обычно же исполняющая система запускается с помощью ярлыка прямо с рабочего стола Windows. Далее будет использоваться в качестве примера исполняющая система RunAnimaWF.exe.

Сначала создаём на рабочем столе ярлык для файла RunAnimaX.exe. После создания ярлыка нужно изменить его свойства, выбрав у ярлыка пункт «Свойства» в контекстном меню:



Сразу после имени программы RunAnimaWF.exe нужно указать имя скомпилированного файла для узла вместе с каталогом проекта. Если каталог или имя узла содержат пробелы, то имя файла нужно заключить в кавычки. Рабочая папка должна содержать имя каталога, к котором расположена исполняющая система RunAnimaWF.exe.

Программа разметки медиа-файлов

Проигрыватель AnimaPlayer.exe предназначен для разметки видео и аудиофайлов. Разметка состоит в том, что медиафайлу сопоставляется список меток — временных точек.

Окно проигрывателя состоит из области просмотра, полоски позиции, панели управления и списка меток. Панель управления содержит командные кнопки, два индикатора времени (общее и текущее время в формате час:минута:секунда.фрейм) и регулятор громкости. В правой части окна находится список меток. Метки сохраняются в файле, который имеет то же имя, что и медиафайл, но расширение lst.

Команды управления:

 Открыть медиафайл. По этой команде отображается диалог выбора файла и выбранный файл открывается. Если медиафайл уже имеет файл списка меток (lst-файл), то он также открывается и метки отображаются в списке. Можно открывать только те медиафайлы, для который в операционной системе установлены кодеки. Без специальных кодеков можно открывать и проигрывать видеофайлы wmv и некоторые avi, а также аудиофайлы wav и некоторые mp3.

 Сохранить файл меток. При закрытии программы файл меток сохраняется автоматически.

 Запустить воспроизведение медиафайла.

 Приостановить проигрывание.

 Остановить проигрывание и перемотать позицию файла в начало.

 Добавить новую метку времени. Метка будет вставлена в список в порядке увеличения времен.

 Удалить метку, выделенную в списке.

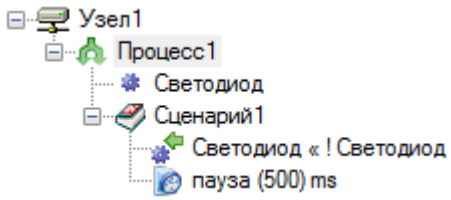
Для прямого перехода к нужной позиции медиафайла можно щелкать мышкой по полоске позиции. Двойной щелчок по метке в списке приведет к позиционированию медиафайла на указанную метку.

Приложение. Примеры

Проекты примеров находятся в каталоге «Projects\Примеры».

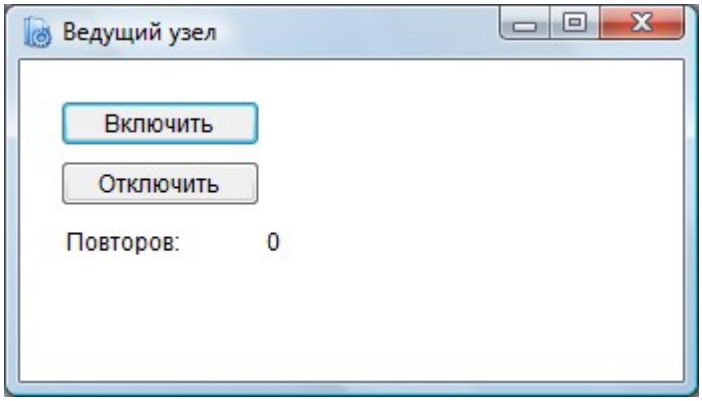
Пример1

Этот пример подробно обсуждался в разделе «Первая программа».

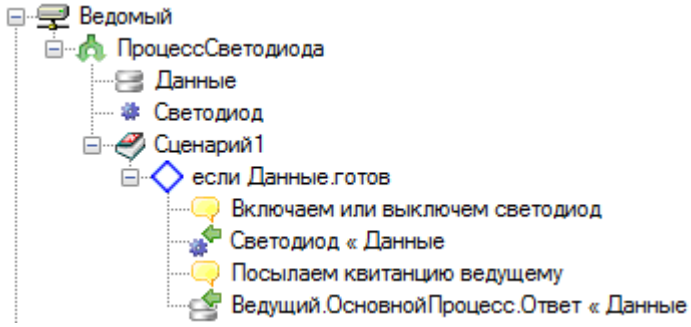


Пример2

В примере показан вариант надежной связи по каналам между двумя узлами. Ведомый узел реализован на микроконтроллере LPC2378, а ведущий – на персональном компьютере и исполняющей системе WF. Узлы соединены сетью RS232. В программе ведущего узла отображается окно, содержащее две кнопки. Первая кнопка включает светодиод на микроконтроллере, вторая — отключает.



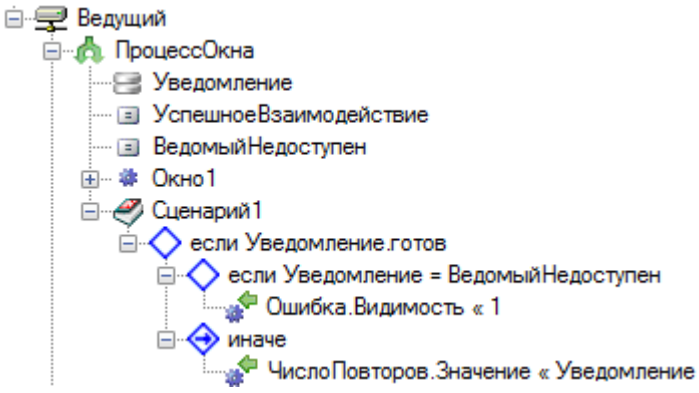
Сценарий ведомого узла следующий: микроконтроллер ожидает появление данных на своем канале и, в зависимости от переданных данных, включает или выключает светодиод и отвечает ведущему узлу подтверждением-квитанцией. Значение квитанции соответствует переданным данным.



Ведущий узел с помощью кнопок посылает сигнал ведомому узлу и ожидает от него квитанции. Если квитанция не получена в течении времени таймаута, то ведущий повторяет передачу 6 раз. Квитанция не засчитывается, если значение квитанции не равно переданным данным — эта ситуация считается такой же ошибкой, как и таймаут. Если все шесть раз передача была неуспешной, то в окне отображается красный прямоугольник, индицирующий ошибку и нажатие на любую кнопку не приводит к каким-либо действиям.

Алгоритм ведущего узла содержит два процесса — один процесс работает с окном и кнопками, а второй процесс занимается взаимодействием с ведомым узлом. Это хороший принцип — разделение отображения и логики работы. Поскольку оба этих процесса работают в одном узле, то необходимости в обеспечении надежной связи между ними нет. Кнопка включения посылает в канал кнопок значение 1, а кнопка выключения — 0.

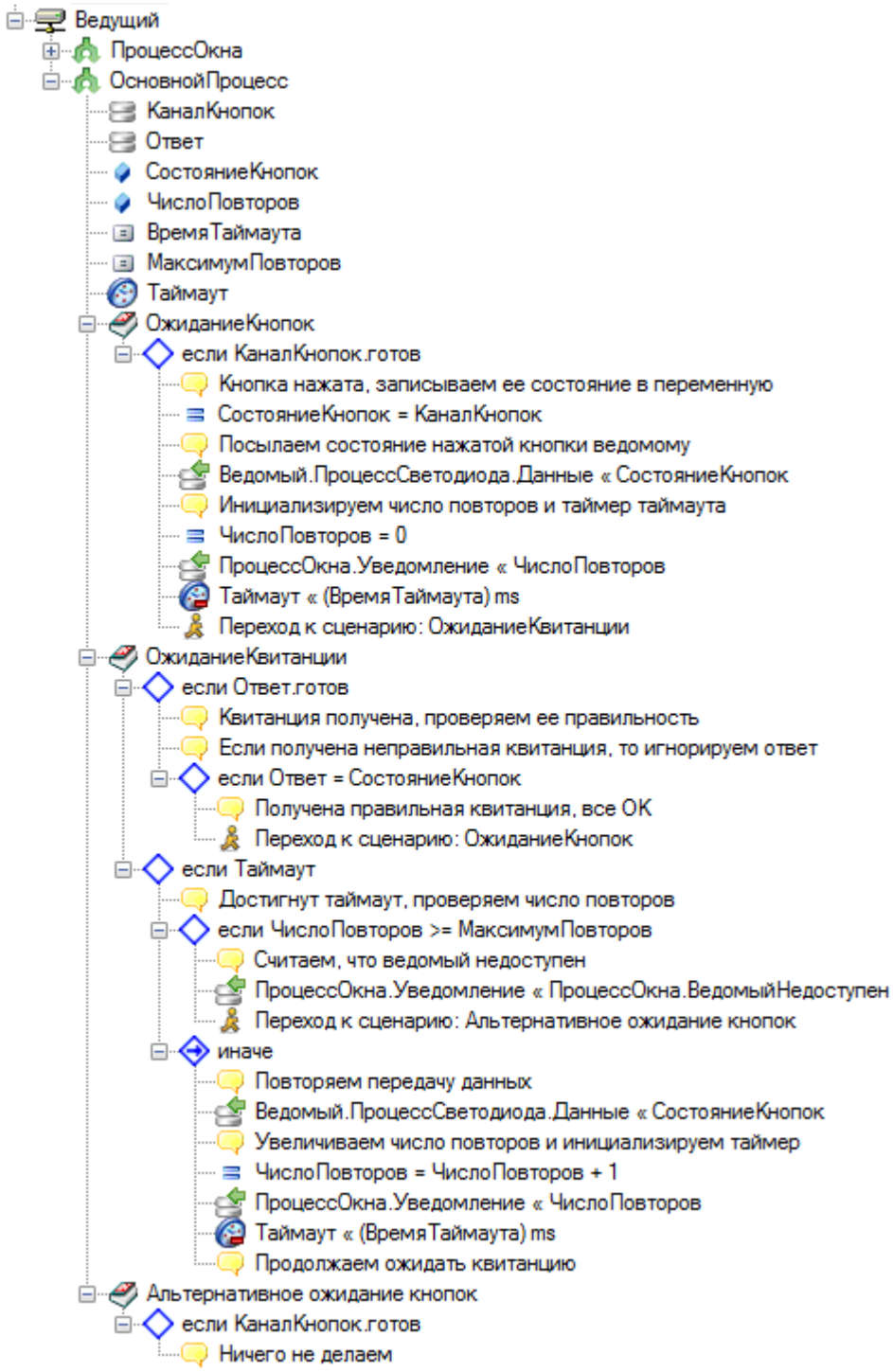
Сценарий в процессе «ПроцессОкна» принимает уведомления от основного процесса о числе повторов и фатальной ошибке, когда ведомый узел недоступен.



Основной процесс имеет три сценария, соответствующие состояниям алгоритма работы — ожидание кнопок, ожидание квитанции и альтернативное ожидание кнопок в случае недоступности ведомого.

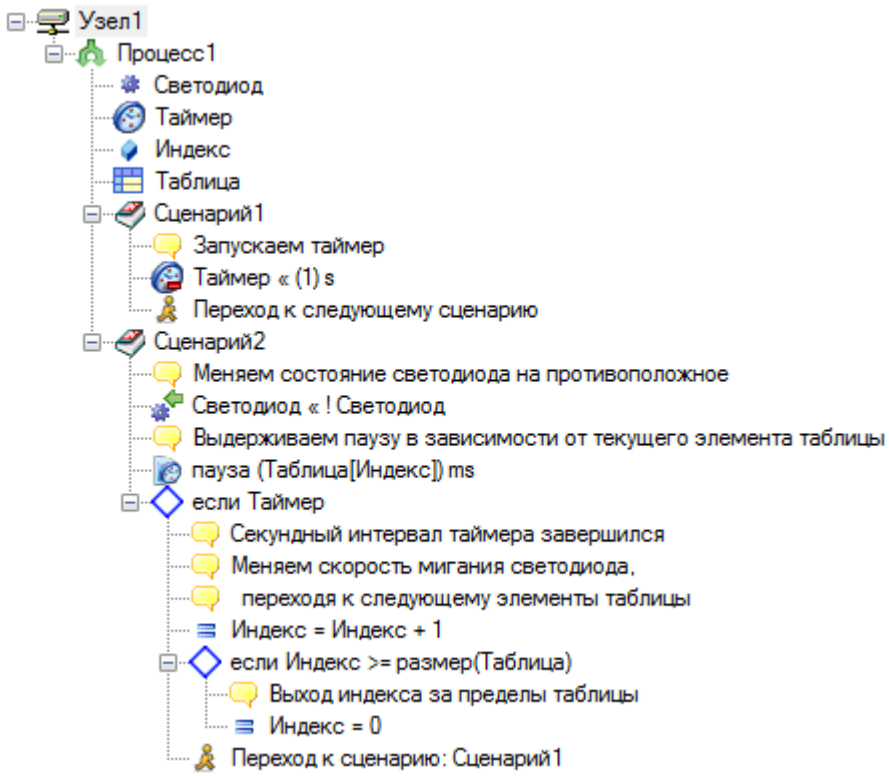
Для запуска программы нужно зашить программу ведомого узла и запустить программу ведущего. Затем нужно проверить нормальную работу, нажимая кнопки включения и выключения. Для имитации отказов можно воспользоваться кнопкой RESET на отладочной плате микроконтроллера. Для демонстрации период повторного взаимодействия при ошибке выбран равным 1 секунде, в реальности же период выбирается в диапазоне от 10 до 100 миллисекунд. Если нажать кнопку RESET и, не отпуская её, нажать кнопку «Включить» в окне ведущего узла, то можно увидеть, что число повторов будет увеличиваться. Если отжать RESET не дожидаясь шестикратного повтора, то взаимодействие завершится успешно и последующие включения-выключения светодиода будут успешными. Если держать RESET нажатой больше шести повторов, то на форме появится индикатор ошибки.

Это достаточно сложный пример, демонстрирующий надежное взаимодействие между узлами и простое взаимодействие между процессами в одном узле. Такую технику нужно использовать только в тех случаях, когда надежность взаимодействия является критичной, а связь между узлами потенциально ненадежна.


Ведущий
ПроцессОкна
ОсновнойПроцесс
КаналКнопок
Ответ
СостояниеКнопок
ЧислоПовторов
ВремяТаймаута
МаксимумПовторов
Таймаут
ОжиданиеКнопок
если КаналКнопок.готов
Кнопка нажата, записываем ее состояние в переменную
СостояниеКнопок = КаналКнопок
Посылаем состояние нажатой кнопки ведомому
Ведомый.ПроцессСветодиода.Данные « СостояниеКнопок
Инициализируем число повторов и таймер таймаута
ЧислоПовторов = 0
ПроцессОкна.Уведомление « ЧислоПовторов
Таймаут « (ВремяТаймаута) ms
Переход к сценарию: ОжиданиеКвитанции
ОжиданиеКвитанции
если Ответ.готов
Квитанция получена, проверяем ее правильность
Если получена неправильная квитанция, то игнорируем ответ
если Ответ = СостояниеКнопок
Получена правильная квитанция, все ОК
Переход к сценарию: ОжиданиеКнопок
если Таймаут
Достигнут таймаут, проверяем число повторов
если ЧислоПовторов >= МаксимумПовторов
Считаем, что ведомый недоступен
ПроцессОкна.Уведомление « ПроцессОкна.ВедомыйНедоступен
Переход к сценарию: Альтернативное ожидание кнопок
иначе
Повторяем передачу данных
Ведомый.ПроцессСветодиода.Данные « СостояниеКнопок
Увеличиваем число повторов и инициализируем таймер
ЧислоПовторов = ЧислоПовторов + 1
ПроцессОкна.Уведомление « ЧислоПовторов
Таймаут « (ВремяТаймаута) ms
Продолжаем ожидать квитанцию
Альтернативное ожидание кнопок
если КаналКнопок.готов
Ничего не делаем

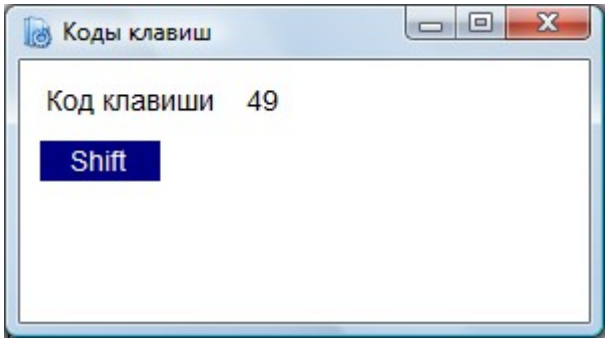
Пример3

Пример демонстрирует использование таблицы для задания нелинейной функции. В этом примере, как и в Примере1 мы заставим мигать светодиод, но скорость его мигания будем меняться каждую секунду в зависимости от таблицы.

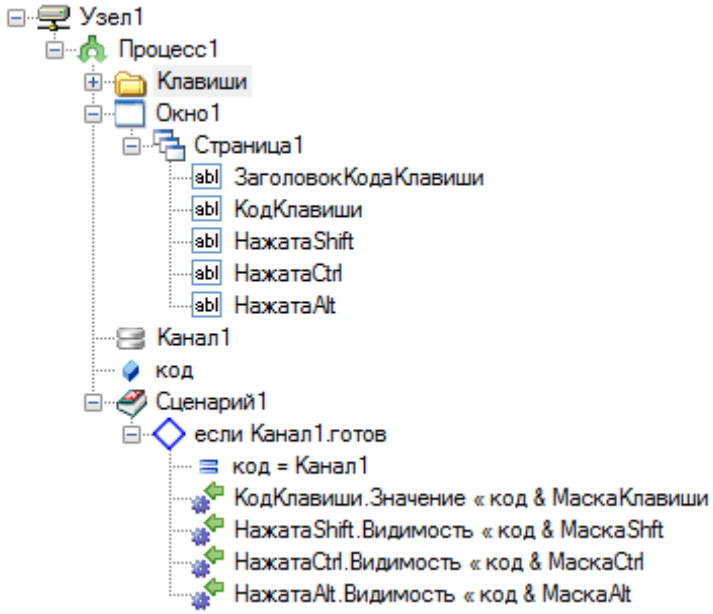


Пример4

Пример демонстрирует использование клавиатуры для процессора WF. Окно программы выглядит так:



Вид окна соответствует нажатым клавишам Shift и 1.

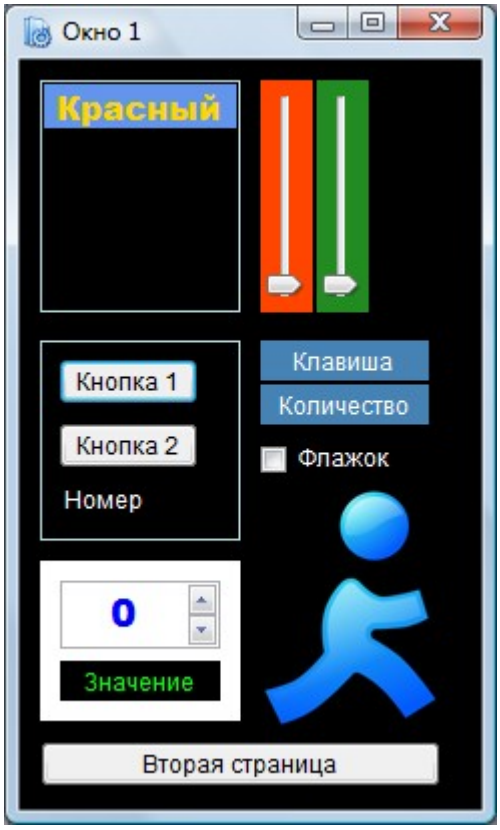


Группа «Клавиши» на показанной программе не раскрыта — она содержит длинный список констант — кодов клавиш и кодов масок. Маски позволяют выделять из принятого кода собственно код клавиши и признаки клавиш-модификаторов (Shift, Ctrl, Alt). Можно копировать всю группу в другие проекты для того, чтобы использовать названия клавиш вместо их кодов.

Окно направляет коды клавиш в Канал1. Сценарий определяет готовность канала и получает из него код клавиши. Значение кода отображается в устройстве КодКлавиши, а состояние клавиш-модификаторов влияет на видимость соответствующих устройств НажатаShift, НажатаCtrl, НажатаAlt.

Пример5

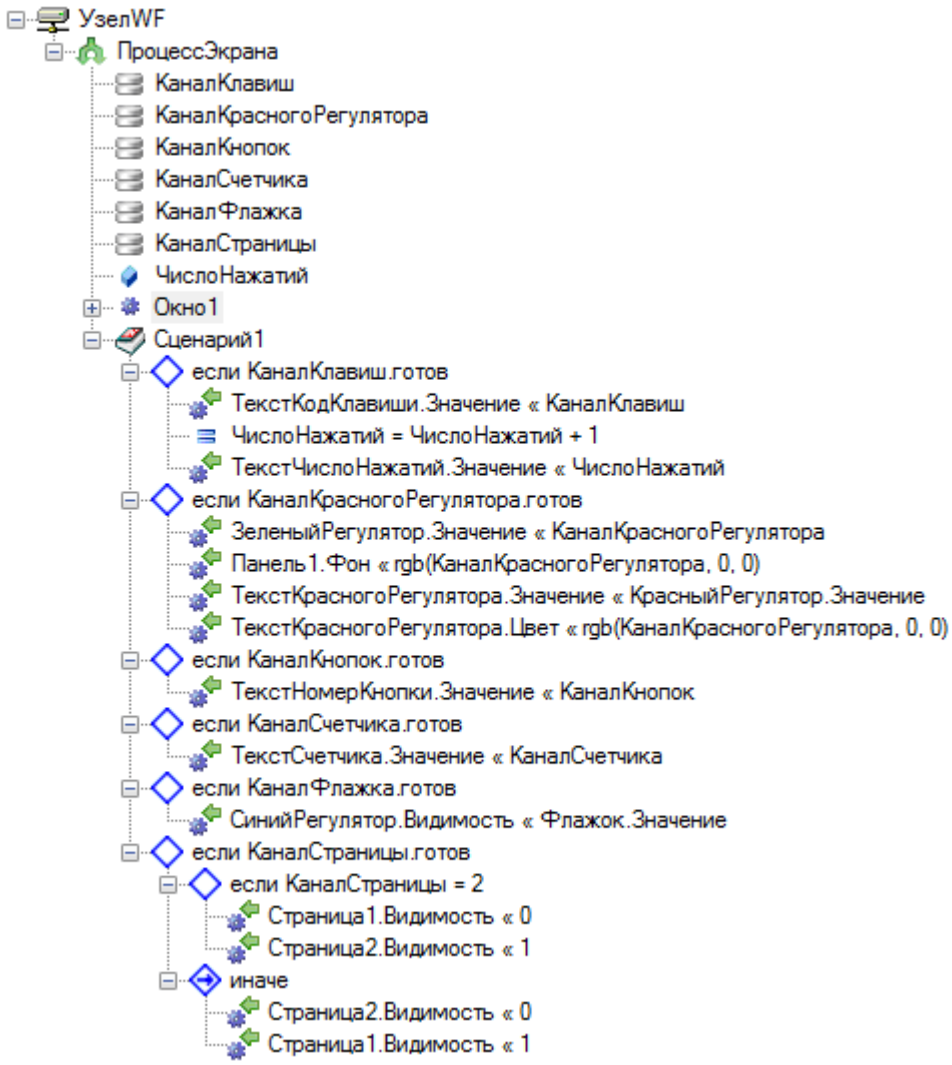
Пример демонстрирует все устройства процессора WF. Внешний вид окна:



Окно содержит две рамки. Поскольку специального устройства-рамки нет, рамка делается вложением двух панелей — одна панель имеет цвет рамки, а другая — черный цвет. Ширина рамки определяется соотношением координат верхней и нижней панели. Верхняя рамка содержит устройство-текст, которое отображает значение красного регулятора. Средняя рамка содержит две кнопки и текст, отображающий номер нажатой кнопки. Нижняя панель содержит счетчик и текст, дублирующий значение счетчика при его изменении.

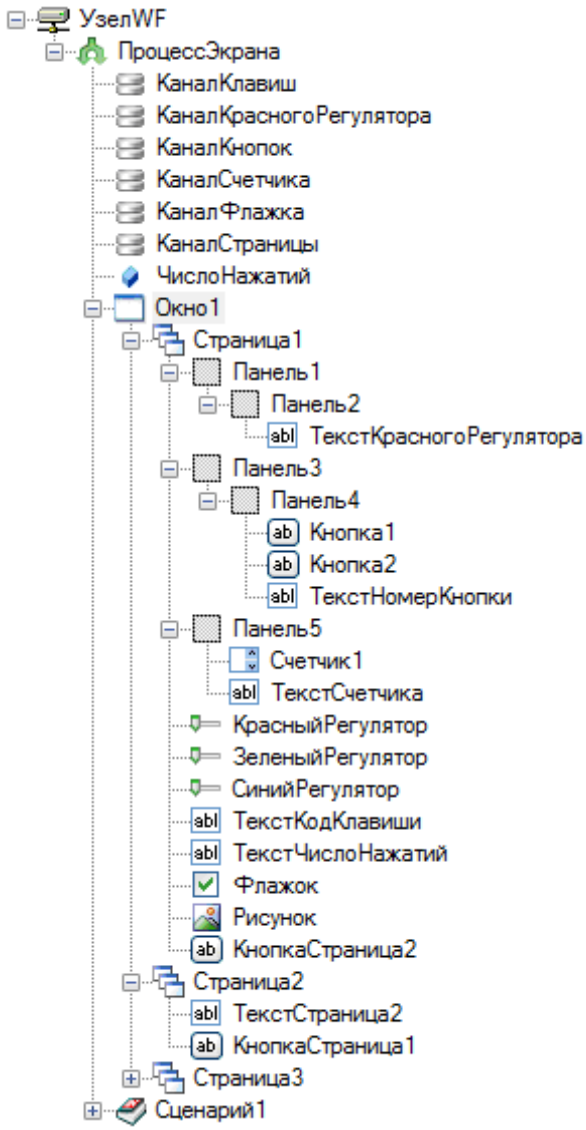
Справа в окне расположены три регулятора — красный, зеленый и синий. Синий регулятор становится видимым только при включении флажка. Изменение значений красного регулятора отображаются в верхней рамке и дублируются в зеленом регуляторе. Изменение красного регулятора также изменяет цвет рамки и текста значения.

Два устройства-текста (клавиша и количество) отражают код последней нажатой клавиши и число нажатий. Графический рисунок реагирует на щелчок мышки — при этом устройство-текст счетчика получает значение 88. Программа узла:



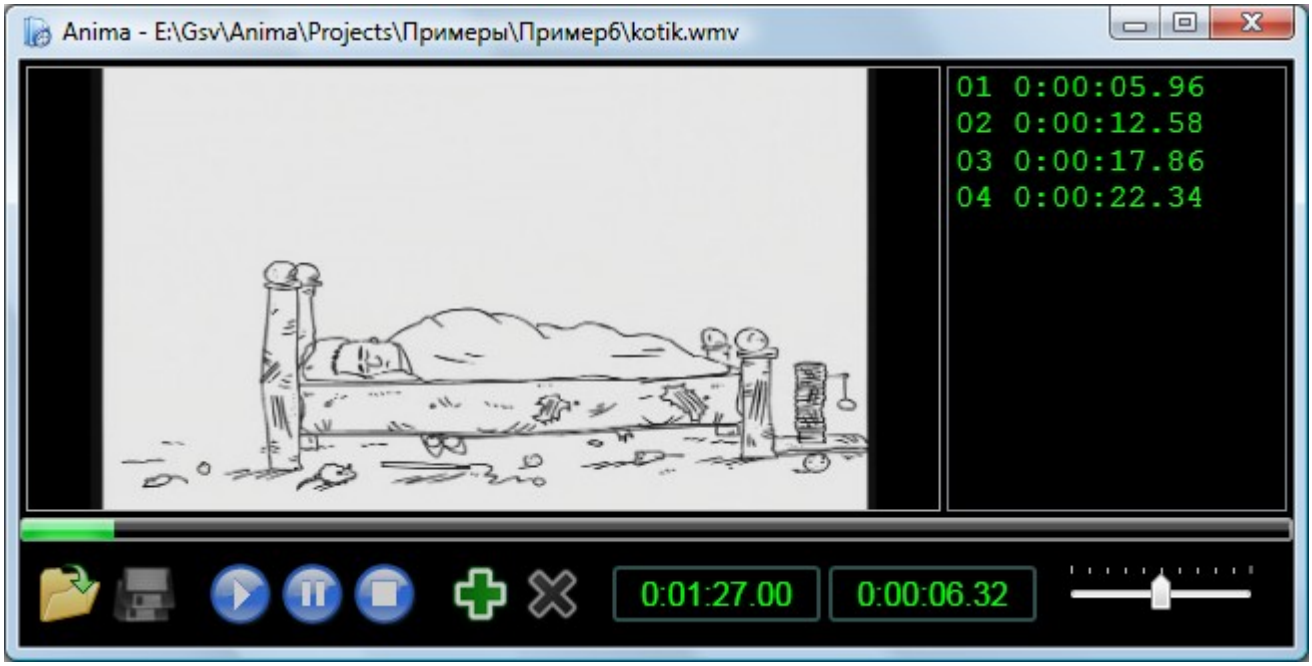
Переключение на вторую страницу выполняется соответствующей кнопкой. В сценарии это достигается тем, что одна страница скрывается, а другая отображается.

На следующем рисунке показана структура окна, которая была свернута в предыдущем рисунке:

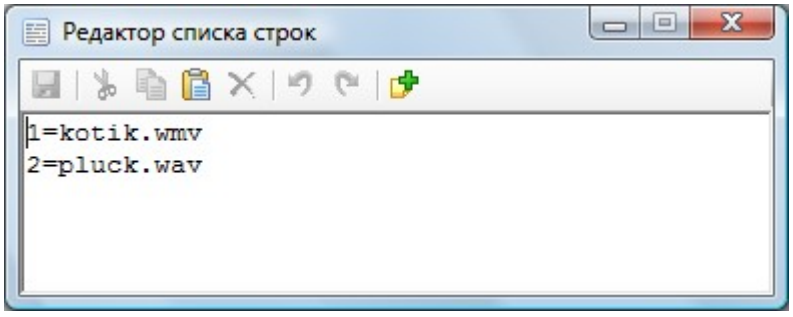

УзелWF
ПроцессЭкрана
КаналКлавиш
КаналКрасногоРегулятора
КаналКнопок
КаналСчетчика
КаналФлажка
КаналСтраницы
ЧислоНажатий
Окно1
Страница1
Панель1
Панель2
ТекстКрасногоРегулятора
Панель3
Панель4
Кнопка1
Кнопка2
ТекстНомерКнопки
Панель5
Счетчик1
ТекстСчетчика
КрасныйРегулятор
ЗеленыйРегулятор
СинийРегулятор
ТекстКодКлавиши
ТекстЧислоНажатий
Флажок
Рисунок
КнопкаСтраница2
Страница2
ТекстСтраница2
КнопкаСтраница1
Страница3
Сценарий1

Пример6

Пример демонстрирует устройства процессора WPF. После запуска программы в окне отображается размеченный видеофайл и при каждом событии видеофайла (открытие, закрытие файла и временная метка) воспроизводится короткий аудиофайл. Используется следующая разметка видеофайла:



Также используется следующий список строк для программы узла:



Файл с индексом 1 — это видеофайл, а 2 — аудиофайл. Так как путь файлов явно не указан, это означает, что файлы расположены в каталоге проекта «Пример6».

Программа примера: