Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

А.Е. Карелин
Д.А. Рождественский
Т.Е. Григорьева
Ю.А. Шурыгин

РАЗРАБОТКА ПРОЕКТА АВТОМАТИЗАЦИИ

В SCADA-СИСТЕМЕ APDAR

методические указания по выполнению практических работ
и организации самостоятельной работы

Томск 2026

УДК 681.5, 004.4
ББК 32.965, 32.973
К22

Рецензент:
Чириков С.В., генеральный директор АО «ЭлеСи»

Карелин, Алексей Евгеньевич
К22 Разработка проекта автоматизации в SCADA-системе APDAR: методические указания по выполнению практических работ и организации самостоятельной работы / А.Е. Карелин, Д.А. Рождественский, Т.Е. Григорьева, Ю.А. Шурыгин − Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2026. – 65 с.

В методических указаниях изложены практические аспекты создания проекта автоматизации: от настройки окружения и конфигурационных файлов до построения графического интерфейса оператора, программирования логики управления и обработки аварийных сообщений, что позволяет самостоятельно разработать прототип системы управления типовым технологическим объектом в отечественной среде APDAR.

Методические указания предназначены для студентов технических вузов.

Авторами данных методических указаний являются профессорско-преподавательский состав кафедры КСУП (А.Е. Карелин, Т.Е. Григорьева, Ю.А. Шурыгин), а также сотрудник профильного предприятия, директор ООО «Автоматизация Производств», г. Томск, Д.А. Рождественский.

Одобрено на заседании кафедры КСУП, протокол № 8 от 12.05.2026 г.

УДК 681.5, 004.4
ББК 32.965, 32.973

Оглавление

Введение

Современные промышленные предприятия все чаще сталкиваются с необходимостью автоматизации технологических процессов. Ключевую роль в решении этой задачи играют SCADA-системы, которые представляют собой программные пакеты для сбора, обработки, отображения и архивирования информации об объекте управления, а также для формирования управляющих воздействий. Одной из таких систем, активно внедряемой в отечественной промышленности, является APDAR.

Целью методических указаний является получение практических навыков создания проекта автоматизации: от первоначальной настройки и конфигурирования до построения графического интерфейса оператора, программирования логики управления и обработки аварийных сообщений.

В методических указаниях последовательно рассматриваются следующие вопросы:

- создание и запуск проекта, структура менеджеров APDAR;
- работа с редактором Para: типы и точки данных, конфигурация элементов, настройка архивов, алармов и периферийных адресов;
- использование графического редактора GEDI: рисование мнемосхем, настройка динамических свойств (поворот, цвет, видимость), создание кнопок управления;
- скриптовое программирование на встроенном языке APDAR: функции dpGet/dpSet, механизм подписки dpConnect, написание глобальных скриптов для имитации поведения оборудования;
- повторное использование компонентов: шаблоны панелей, доллар-параметры, вызов панелей с подстановкой точек данных;
- построение рабочего места оператора: топология панелей, навигация, вход в систему;
- настройка и отображение аварийных сообщений, квитирование.

Все разделы сопровождаются подробными пошаговыми инструкциями и скриншотами, что позволяет обучающемуся выполнять практические работы в режиме «следуй за инструкцией». В результате освоения материала студент приобретает базовые компетенции, необходимые для самостоятельной разработки проектов автоматизации средней сложности в SCADA-системе APDAR.

Методические указания могут использоваться как для аудиторной работы под руководством преподавателя, так и для самостоятельного изучения. Предполагается, что перед началом работы установлена лицензионная или учебная версия APDAR, а также имеются базовые навыки работы с операционной системой Windows.

1. РАЗРАБОТКА ПРОЕКТА В СИСТЕМЕ APDAR

1.1.Создание проекта

Для того, чтобы создать проект необходимо запустить администратор проектов. Для этого заходим в Пуск -> APDAR -> APDAR Администратор проектов. Либо запускаем его по иконке на рабочем столе.

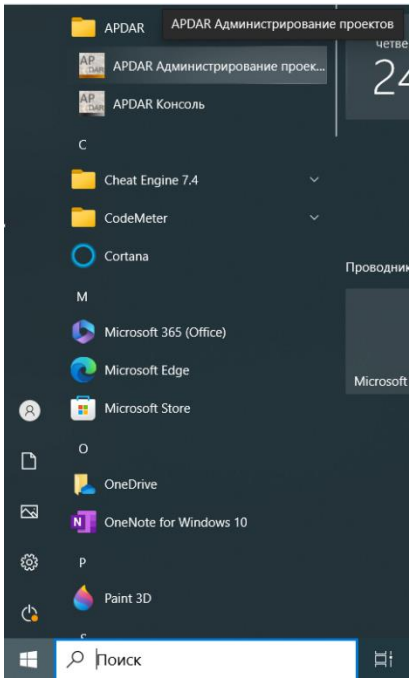


Рисунок 1. Запуск администратора проектов

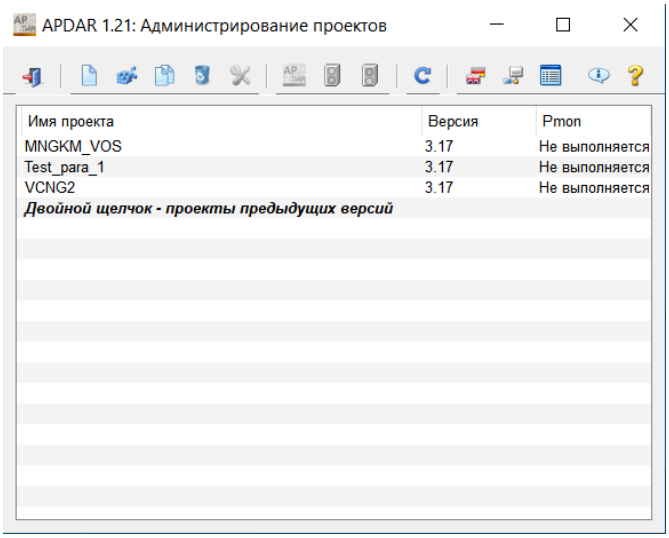


Рисунок 2. Администратор проектов

Панель администрирования проектов позволяет управлять большим количеством проектов: в виде таблицы отображаются текущие зарегистрированные проекты и статус соответствующего PMON (Process Monitor).

Создайте новый проект. Для этого нажмите на изображение чистого листа в панели инструментов или сочетание клавиш Ctrl + N.

Выберите Стандартный проект и нажмите «Далее».

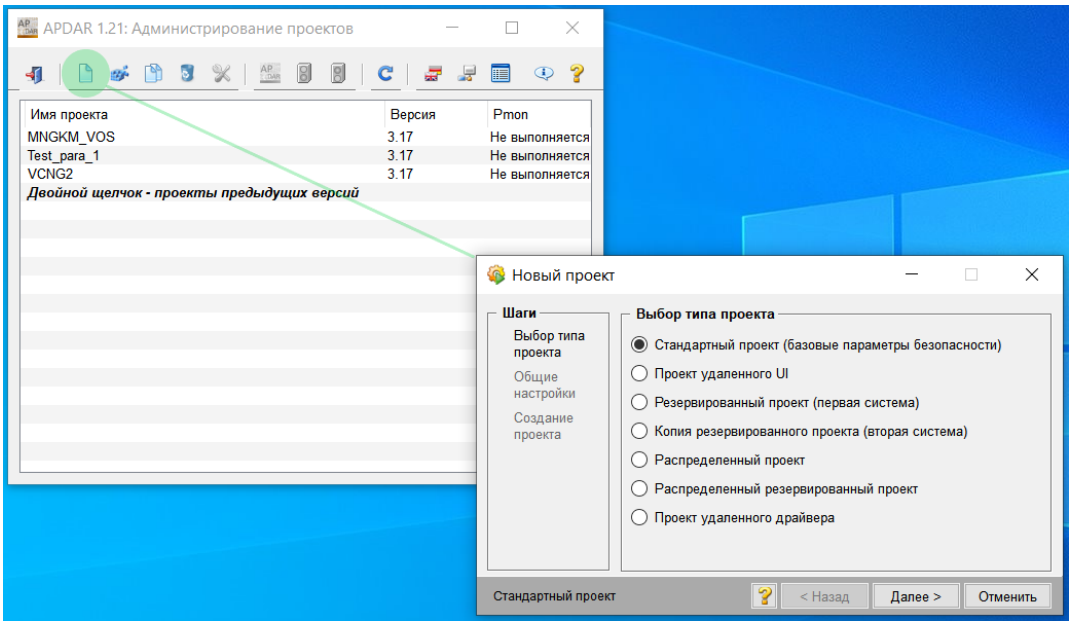


Рисунок 3. Создание нового проекта

В следующем окне латинскими буквами укажите имя проекта и путь до проекта. Избегайте русских букв, пробелов и специальных символов, кроме цифр и знака подчёркивания. Оставляем галочку «Исполняемый». В качестве языков проекта выделите русский и английский. Если планируется многоязычный проект, то набор языков необходимо определить на этом этапе. Нажмите «Далее».

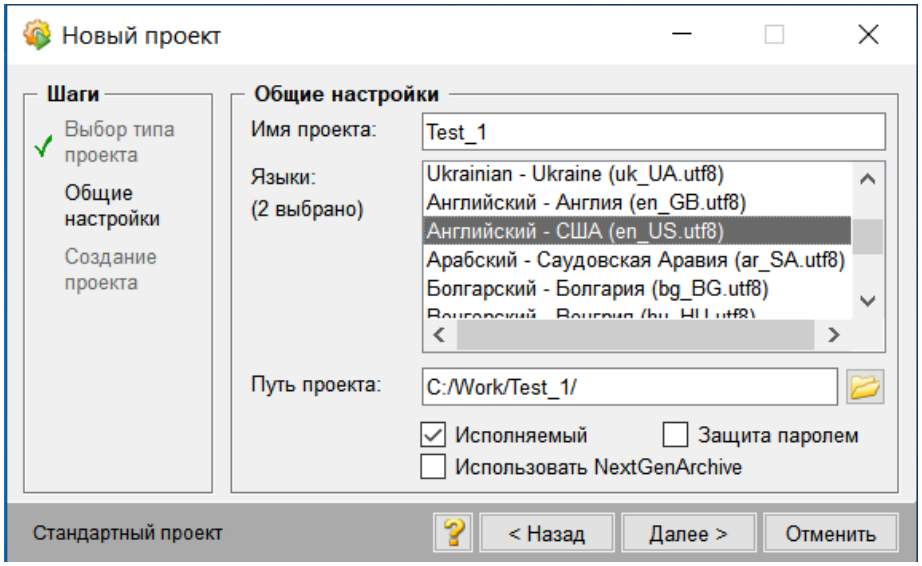


Рисунок 4. Задание имени проекта и пути

Далее нажмите «Ок» и откажитесь от задания пароля суперпользователя (root).

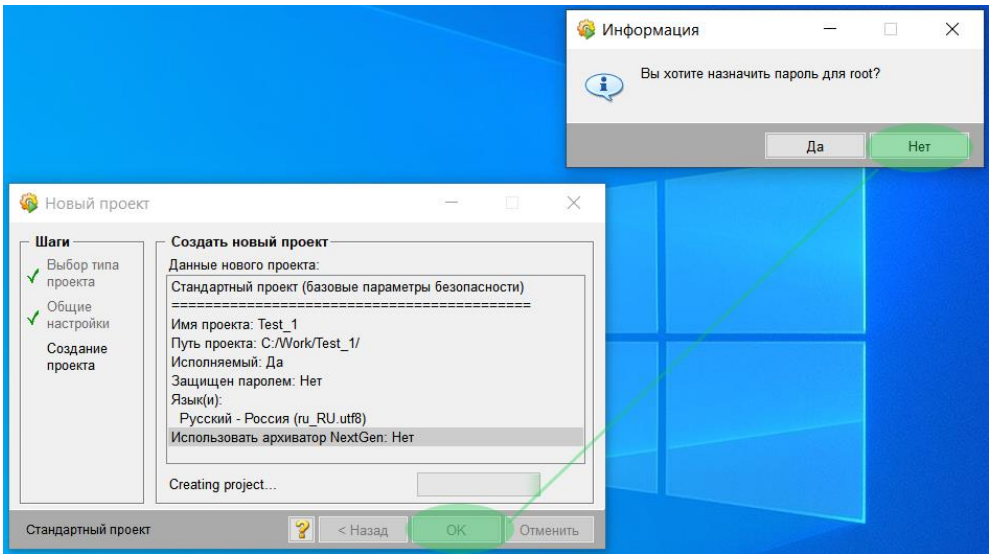


Рисунок 5. Подтверждение настроек и отказ от пароля

Через несколько секунд появится сообщение об успешном создании проекта. Нажмите «Ок».

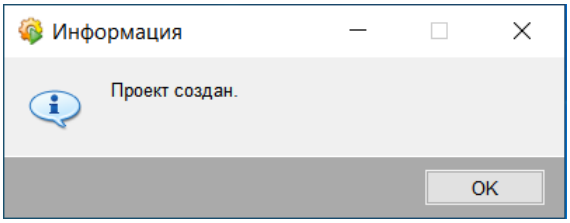


Рисунок 6. Проект создан

Вновь созданный проект отобразится в таблице администратора проектов.

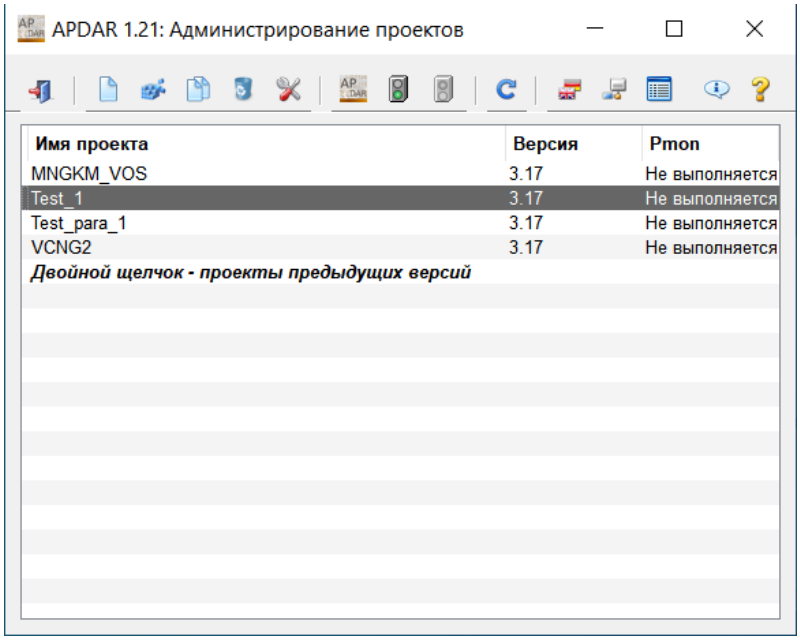


Рисунок 7. Созданный проект в списке проектов

1.2.Запуск проекта

Для запуска проекта в окне администратора проектов выделите свой проект и нажмите на иконку слева от «светофора». Откроется консоль и окно просмотра журналов (в этом окне будут отображаться любые сообщения от системы или менеджеров).

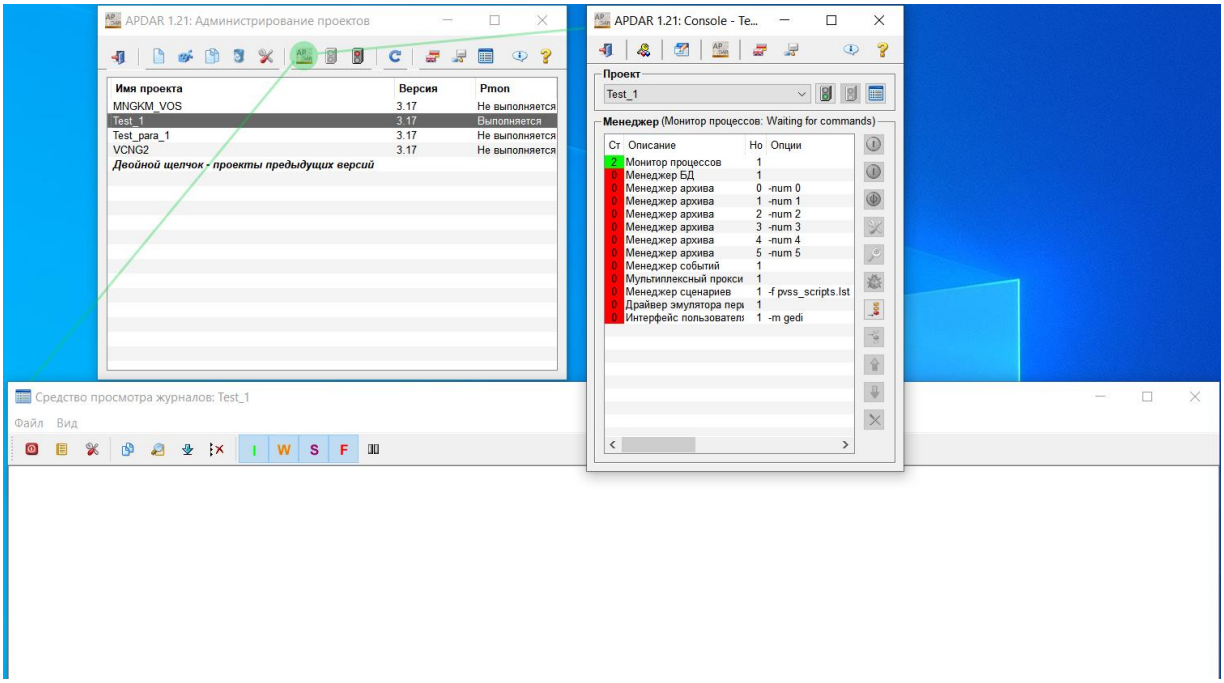


Рисунок 1. Запуск проекта

В окне консоли приведён состав и состояние менеджеров нашего проекта. Здесь можно управлять менеджерами: добавлять, исключать, запускать, останавливать и т.д.

Концепция менеджеров APDAR основана на концепции клиент-сервер. *Event Manager (EV)* и Data Manager (DM) действуют как серверы, другие менеджеры действуют как клиенты.

Менеджер *DM* отвечает за базу данных. Он запускается первым и содержит архивные данные/значения (архивирование), последние значения, аварийные сообщения и структуру *DP* (data point, описание будет далее). Работает с жёстким диском, не работает с реальным временем.

Менеджер *EV (Event Manager)* работает в режиме реального времени, хранит все переменные в оперативной памяти, никогда не обращается к жёсткому диску.

Менеджер *Archive Manager* отвечает за тренды и архивы средствами встроенной в APDAR базы данных. У каждого менеджера одного типа должен быть свой номер, который присваивается путём передачи аргумента –num в строке запуска.

Control Manager. Контрол — это менеджер обработчик скриптов.

Менеджер *UI (User Interface Manager)* пользовательского интерфейса : RT-графика, средства разработки (GEDI и PARA).

В правой части окна (консоль) находятся кнопки управления менеджерами. Слева от имени менеджера находится численный индикатор его состояния.

Состояния:

0: менеджер не запущен;

1: менеджер запускается;

2: менеджер в работе;

3: сбой менеджера / не отвечает более чем через 30 секунд (время ожидания по умолчанию).

Файл конфигурации (config-файл) содержит настройки для проекта APDAR. При необходимости этот файл может быть отредактирован и адаптирован к соответствующим требованиям.

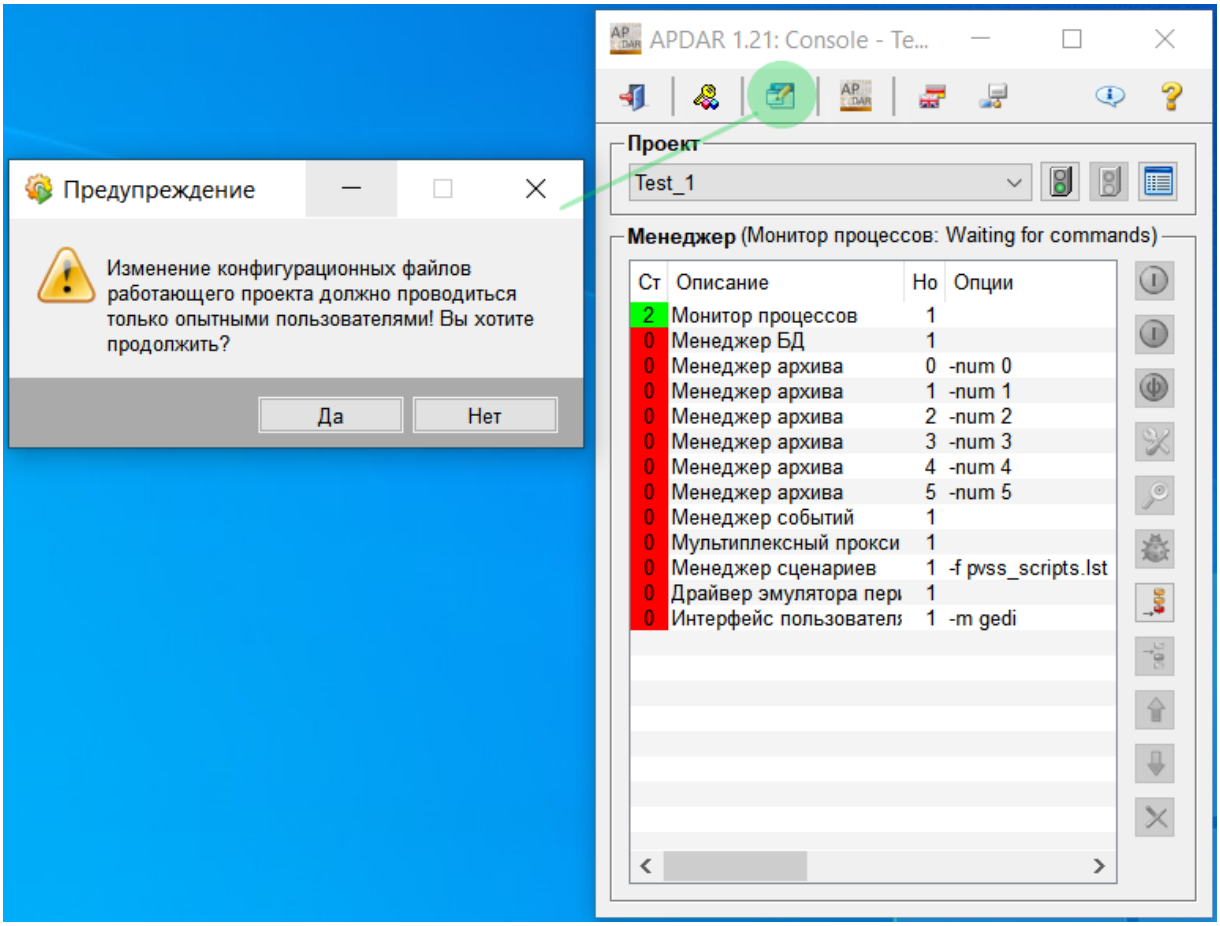


Рисунок 2. Открытие файла конфигурации

Откройте config-файл двойным кликом по имени файла.

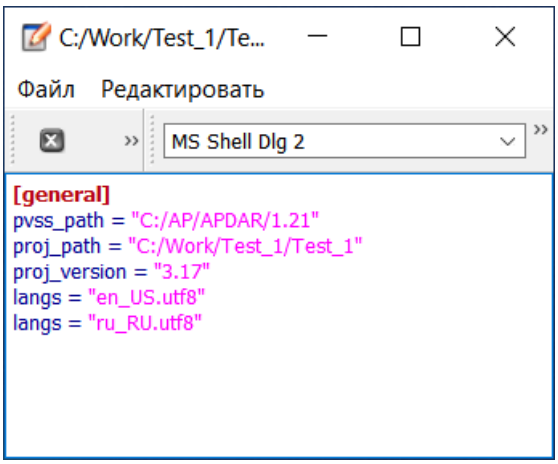


Рисунок 3. Файл конфигурации

Файл разделён на секции (например, [general], [ui] и т.д.), содержащие специальные ключевые слова для настройки различных параметров. Названия разделов указывают, какие менеджеры будут читать их при запуске (например, [general] будет читать все менеджеры).

pvss_path указывает, где установлен APDAR (и расположение каталога bin, в котором находятся все менеджеры).

langs – языки, выбранные для проекта.

ВНИМАНИЕ. Никогда не изменяйте последовательность языковых записей, потому что этот порядок хранится в базе данных. Первый язык всегда является языком по умолчанию в проекте. Если в качестве языка по умолчанию выбран другой язык, это можно сделать следующим образом:

langs: отмечает один из языков проекта как язык проекта по умолчанию. Не требуется, если это первый язык в списке языков или единственный.

ВНИМАНИЕ. Всегда заканчивайте строку в config-файле нажатием клавиши Enter. Если последняя строка не пуста, то последняя с информацией из этого файла не будет прочитана.

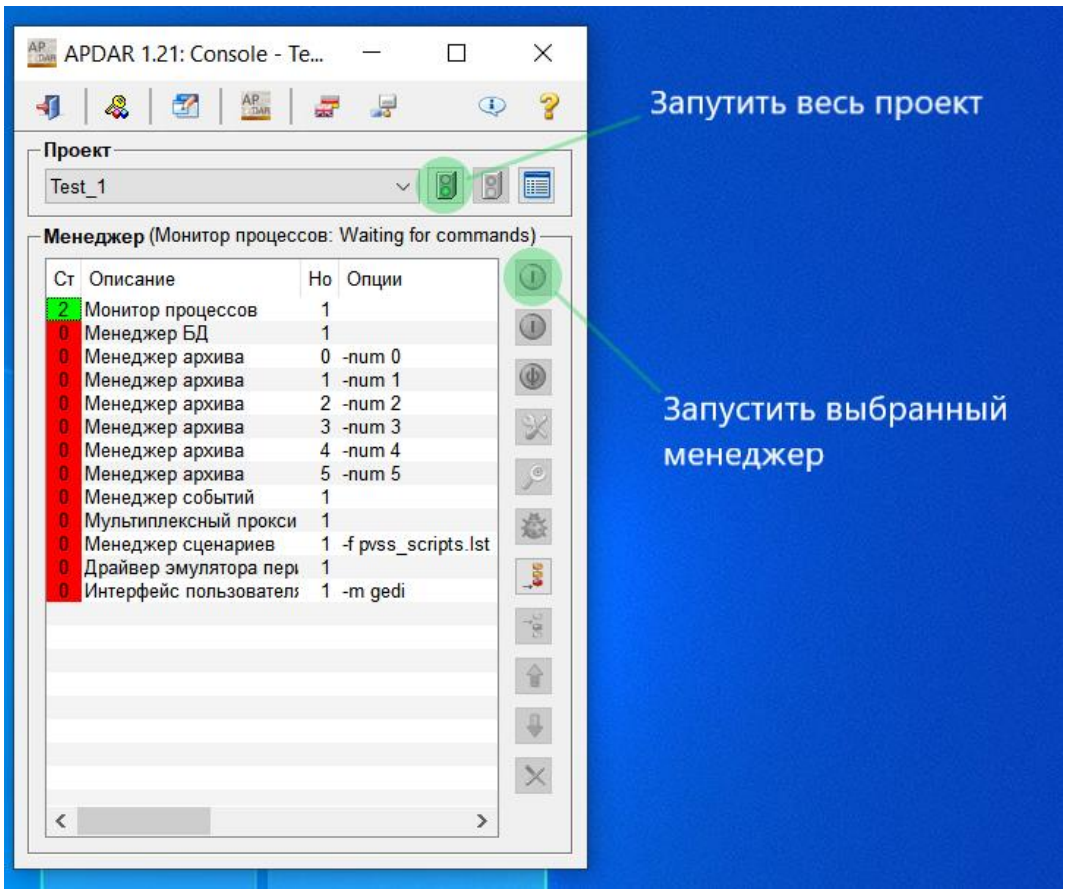


Рисунок 4. Запуск проекта на исполнение

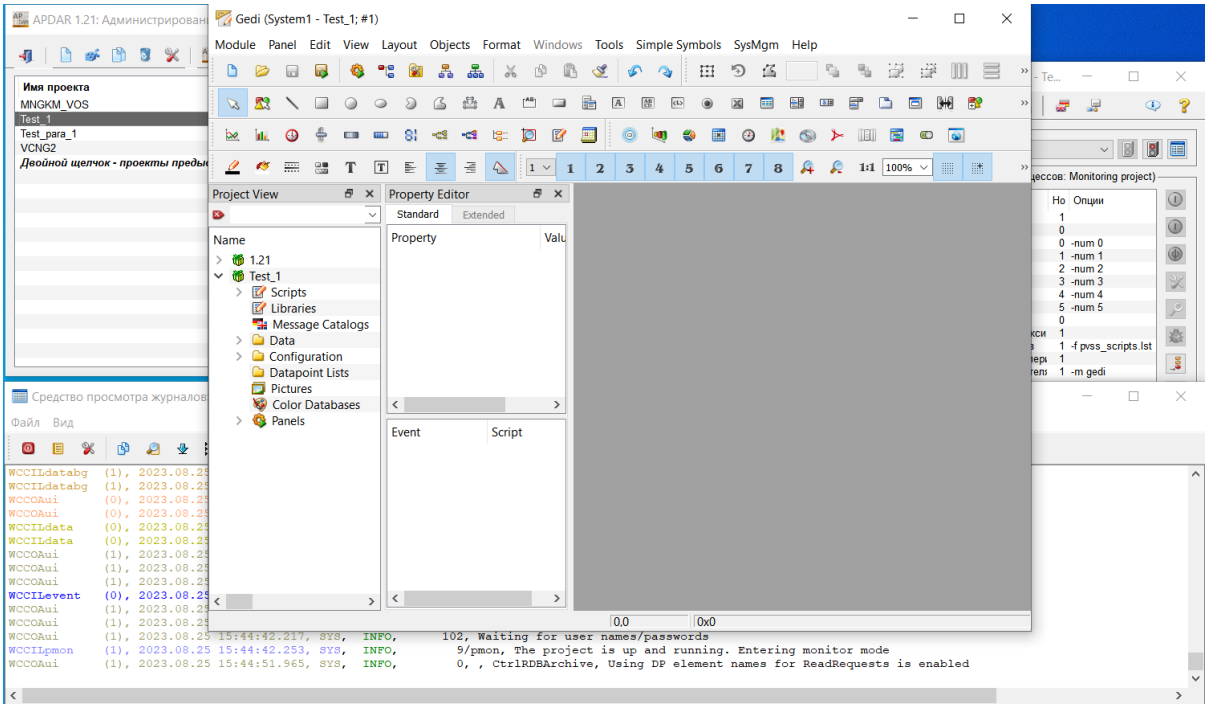


Рисунок 5. Проект запущен

Для нового проекта RT-панель не определена. Система начинается с инструмента разработки GEDI. Здесь мы создадим наш первый проект.

1.3.Редактор Para

Откройте редактор Para. Для этого нажмите кнопку, как показано на рисунке ниже.

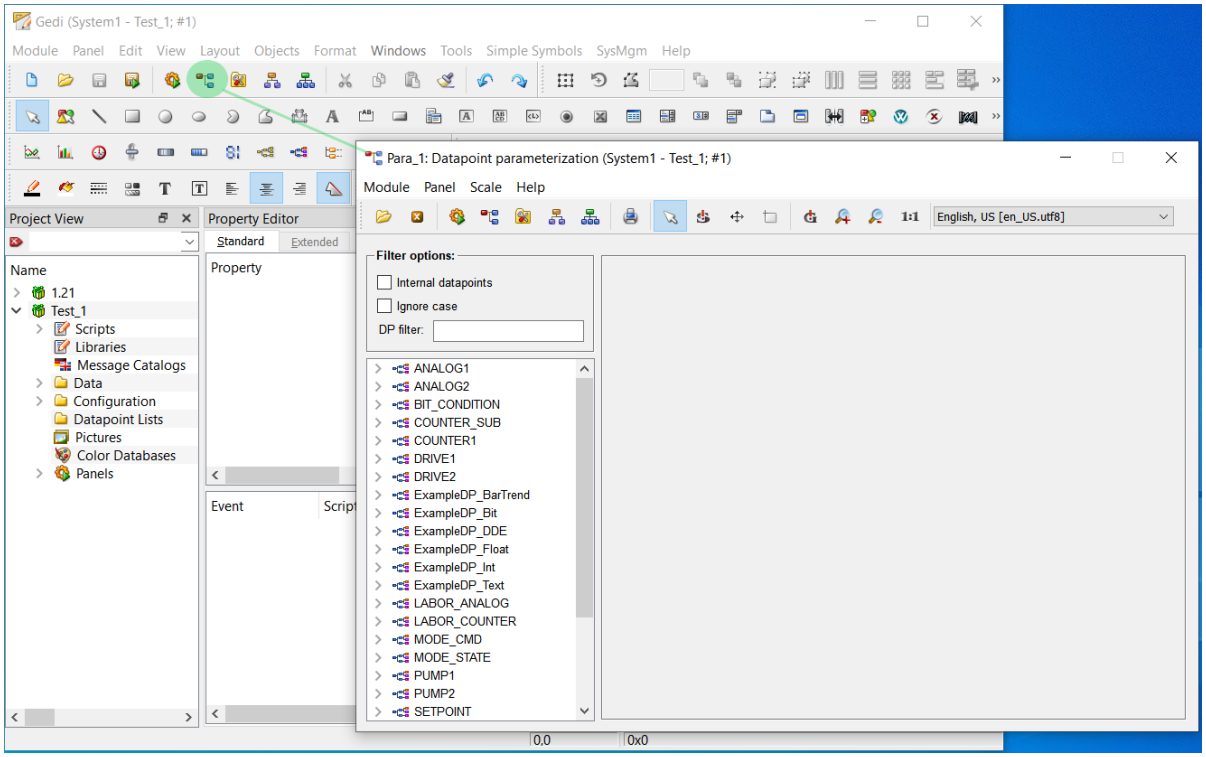


Рисунок 1. Запуск редактора Para

Para предназначен для создания и редактирования типов точек данных и элементов точек данных. Тип точек данных (data point type) — это объявление структуры, а сама точка данных (data point) является непосредственным экземпляром структуры этого типа. Грамотно составить модель данных — это очень важный шаг, и его необходимо продумать еще на начальном этапе.

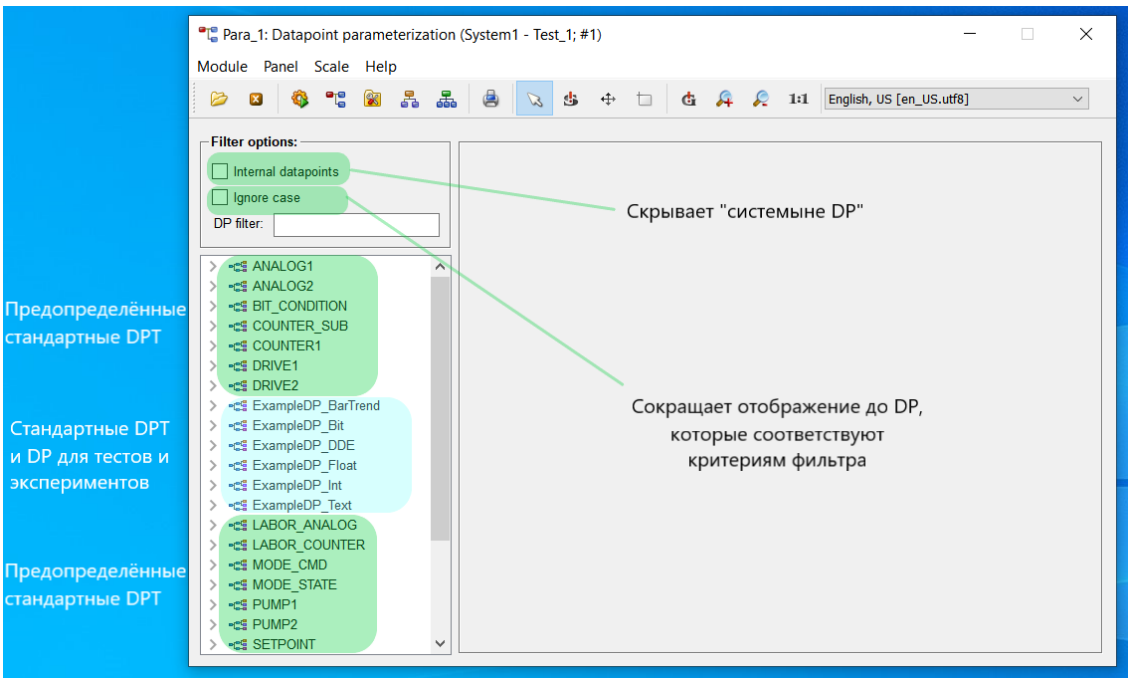


Рисунок 2. Дерево редактора Para

APDAR хранит информацию о состоянии своей системы во «внутренних точках данных». Неправильное обращение с этими DP может привести к повреждению, которое может привести к полному отказу системы. По этой причине внутренние DP по умолчанию скрыты.

Создайте тип точки данных «задвижка» (Valve).

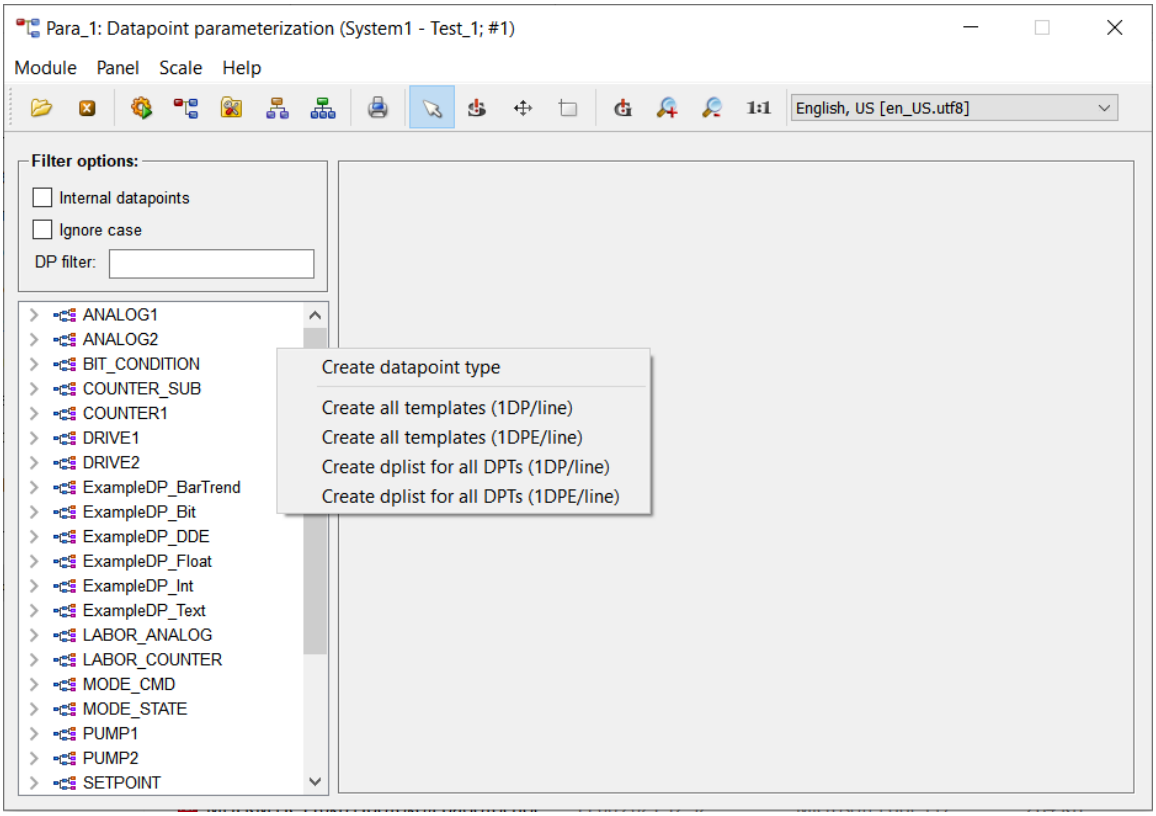
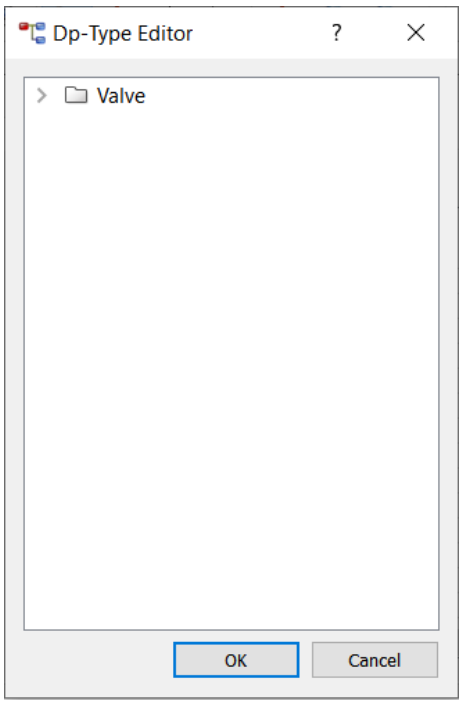
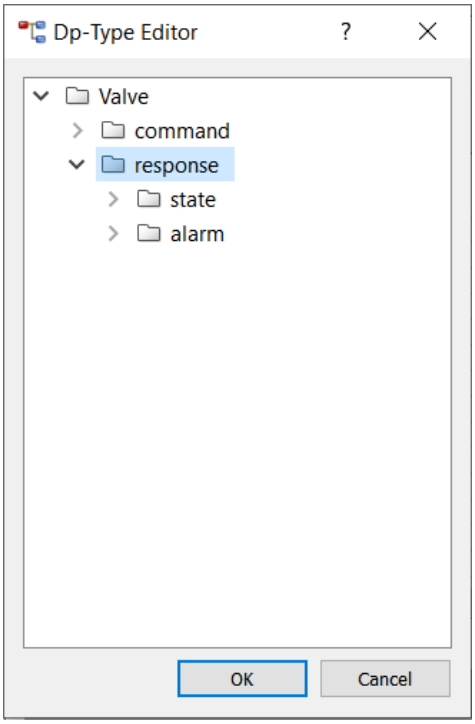


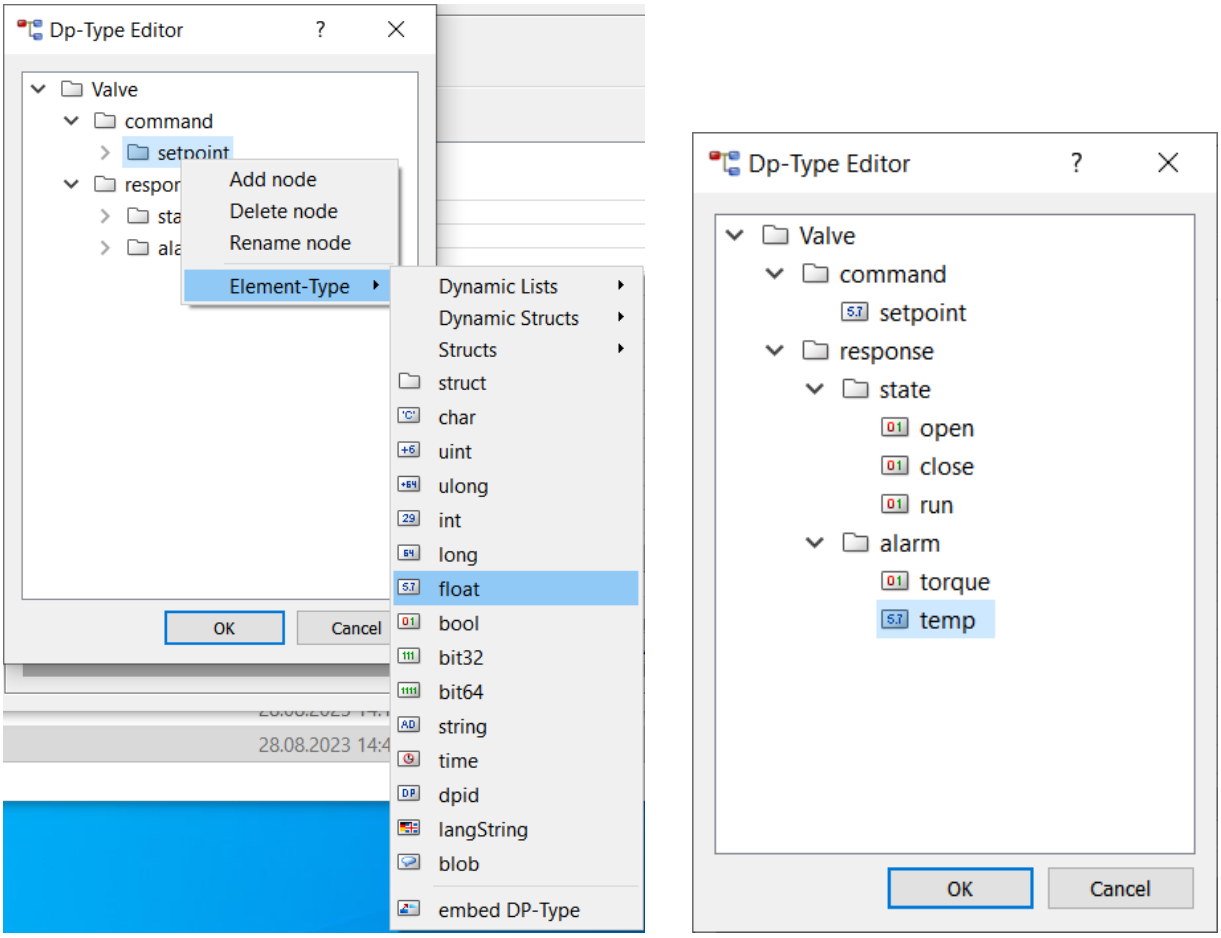
Рисунок 3. Создание типа точки данных



Далее через контекстное меню, вызываемой кликом правой кнопки мыши по имени типа точки данных (Valve) и выбрав пункт меню Add node, создайте следующую структуру.

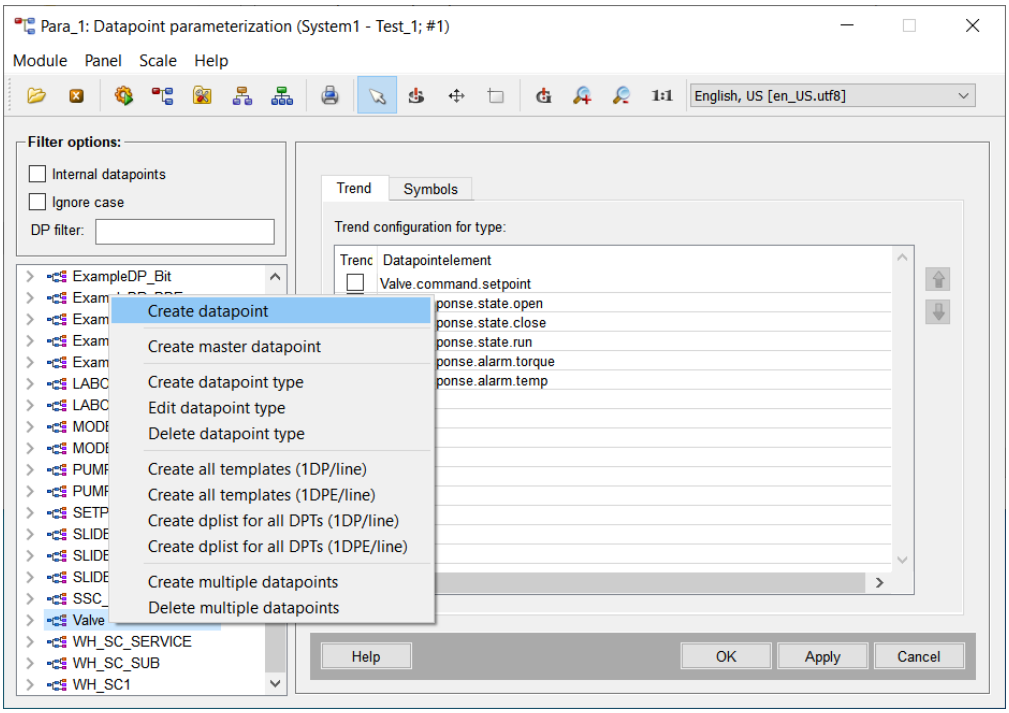


Далее добавьте в дерево типа точки переменные:



Setpoint и temp имеют тип данных float, open, close, run и torque – bool.

Далее создайте точку данных. Для этого в дереве редактора выделите DPT Valve, нажмите правую кнопку мыши и в открышемся контекстном меню нажмите на пункт Create datapoint. Создайте три точки данных (в нашем примере это v1, v2, v3).



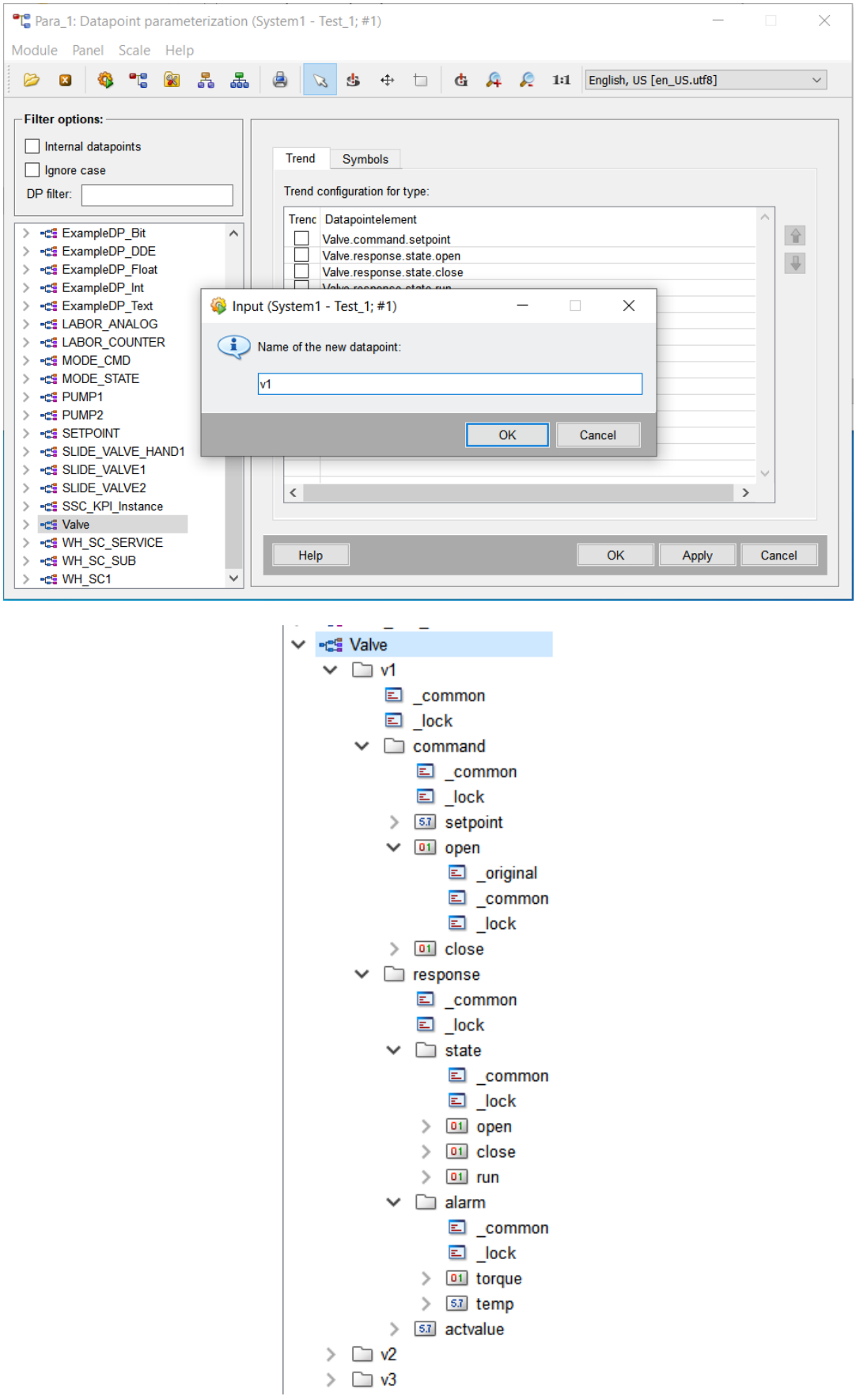


Рисунок 4. Дерево точки данных v1

DP “v1” имеет точно такую же структуру как и DPT “Valve”. Если DPT впоследствии изменится, это изменение передастся каждому DP.

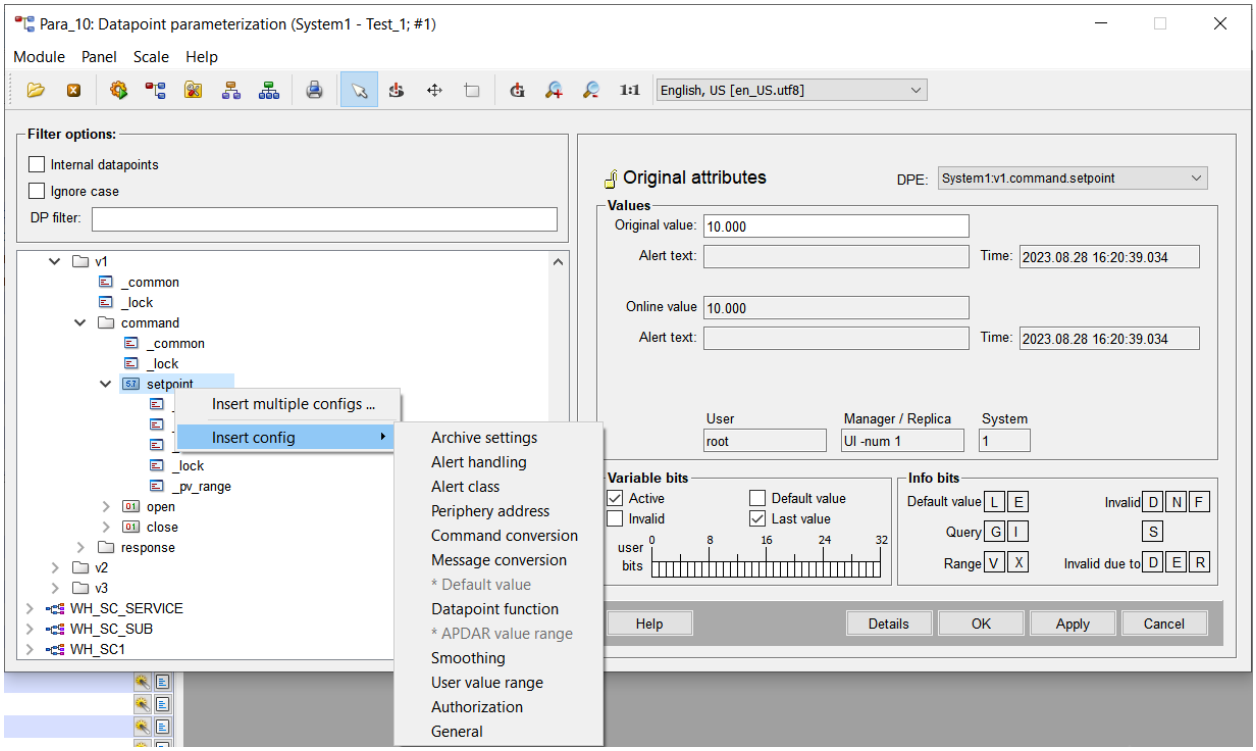
В дереве DP видно, что кроме созданных узлов иерархии и конечных точек появились еще и элементы, которые вы не создавали. Имена элементов начинаются с нижнего подчеркивания: _original, _common и _lock. В системе APDAR они называются конфиги. Каждый конфиг отвечает за какой-то функционал. Это позволяет настраивать каждый DPE (datapoint element) индивидуально.

В зависимости от типа данных атрибуты config могут быть разными.

_oroginal – один из самых главных конфигов DPE. Он содержит значение DPE в системе и его метку времени.

Обратите внимание что поле Original Value доступно для ввода (в настоящий момент времени все эти DPE — внутренние тэги, так как не привязаны ни к какому драйверу), а Online Value — нет. В большинстве случаев эти значения совпадают. Original — это, грубо говоря, то, что «прилетает с поля», а Online — значение переменной в самой системе. Когда они могут не совпадать? Например, если мы реализовали функционал контроля выхода переменной за пределы. Например, если с поля прилетает значение 12 в то время, как максимальное значение — 10, то мы можем задать Online = 10 и выставить бит плохого качества.

Кликните правой кнопкой по DPE setpoint (v1) и посмотрим, какие еще конфиги бывают в системе, и за что некоторые из них отвечают.



Archive settings отвечает за помещение значения в историческую базу данных.

Alert settings — отвечает за журнал тревог и сообщений.

Periphery address — значение переменной берется с драйвера или OPC-сервера и является внешним тэгом. Именно эти DPE считаются в лицензии. То есть, количество

тэгов в проекте — это количество DPE, на которые навесили конфиг «переферийный адрес».

Command conversion и Message conversion — преобразование из инженерных в «физические» (код АЦП, миллиамперы и т.д.) величин и обратно.

Default value и APDAR value range — отвечают за допустимый диапазон значений и подстановку, в случае недостоверности значения.

Smoothing — сглаживание значения аналоговой величины. Весьма важный конфиг с точки зрения производительности системы. Актуален в случае, если драйвер производит постоянный опрос (polling) значения с контроллера. При отсутствии этого конфига каждое новое значение переменной, полученное с ПЛК (например, период опроса для драйвера s7 по умолчанию составляет 100 мс или 10 раз в секунду), будет улетать в сторону EV. Но зачем грузить менеджер событий (как и всю систему) пустой обработкой? В этом случае можно настроить порог, по превышению которого система будет учитывать изменение значения и отдавать его в обработку. В противном случае драйвер продолжает работать тихо, сам по себе, никуда далее сообщения не отсылая.

Authorization — назначение уровня прав доступа к переменной.

Datapoint function — простая математическая обработка значения.

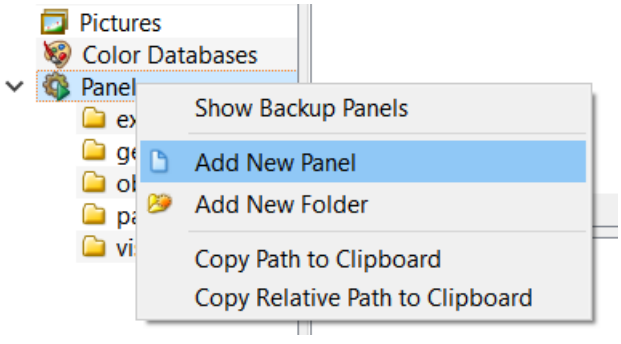
ВНИМАНИЕ. Обратите внимание, что конфиги навешиваются на DPE индивидуально, а не являются частью DPT (типа точки данных).

1.4.Модуль GEDI

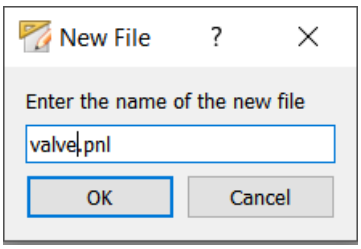
Модуль GEDI отвечает за визуальную составляющую проекта.

Слева в окне Project View вы можете наблюдать имя вашего проекта, а также дерево его составные части.

Обратите внимание на заголовок “Panels”. Определенная структура панелей в проекте уже имеется, но для упрощения мы будем работать в корне. Создайте новую панель, для чего выполните правый клик мышкой по заголовку “Panels” и нажмите пункт меню «Add new panel».

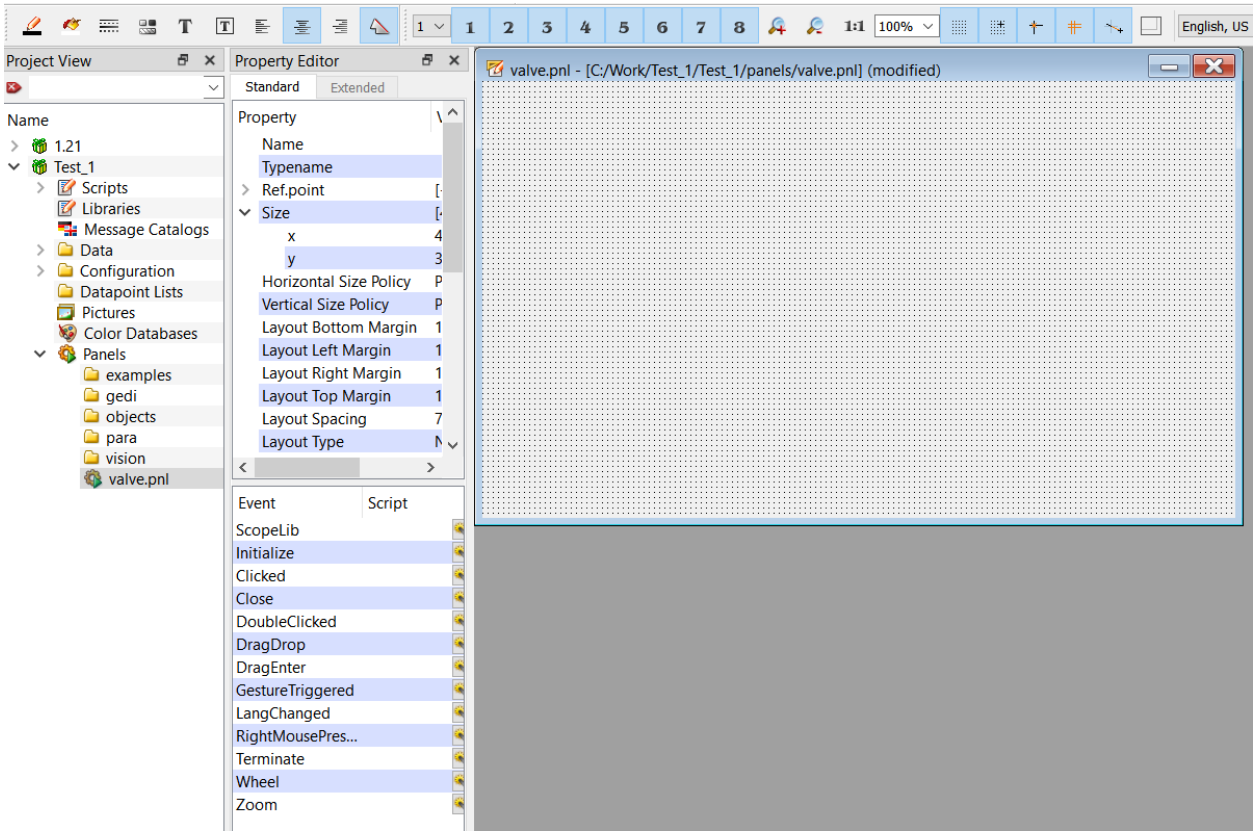


Задайте имя панели “valve”. Нахмите “Ок”.



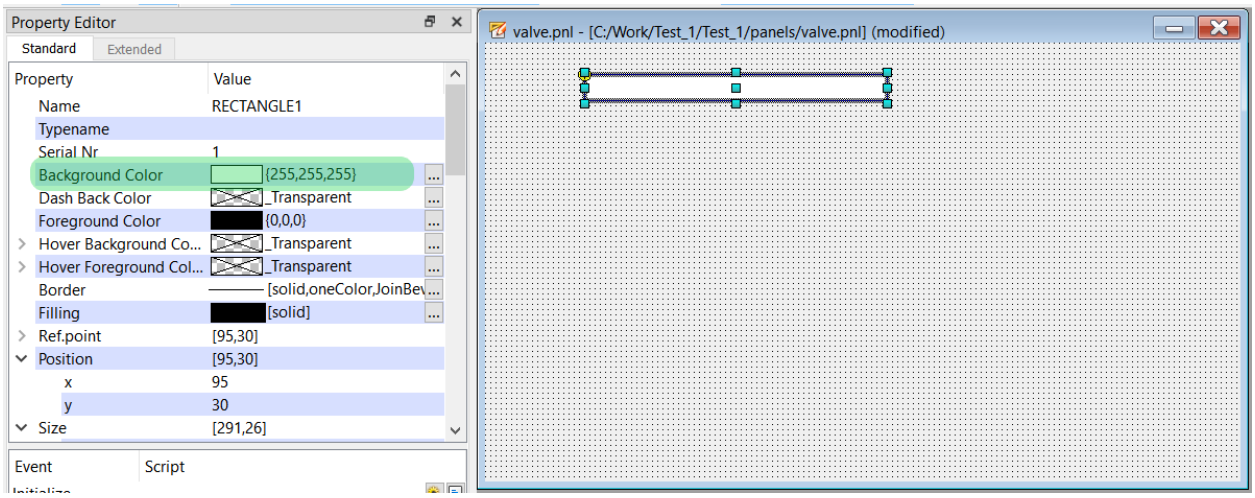
Все «панели» в проекте — это текстовые файлы; они бывают двух типов: pnl или xml. Xml в силу своей структурированности использовать в реальных проектах предпочтительнее, так как его удобнее редактировать в сторонних редакторах. Для изменения формата панели необходимо выполнить «save as» (сохранить как — пункт в меню, потом надо выбрать другой формат данных).

Откройте созданную пустую панель двойным кликом и немного измените ее геометрию, примерно, как на рисунке.

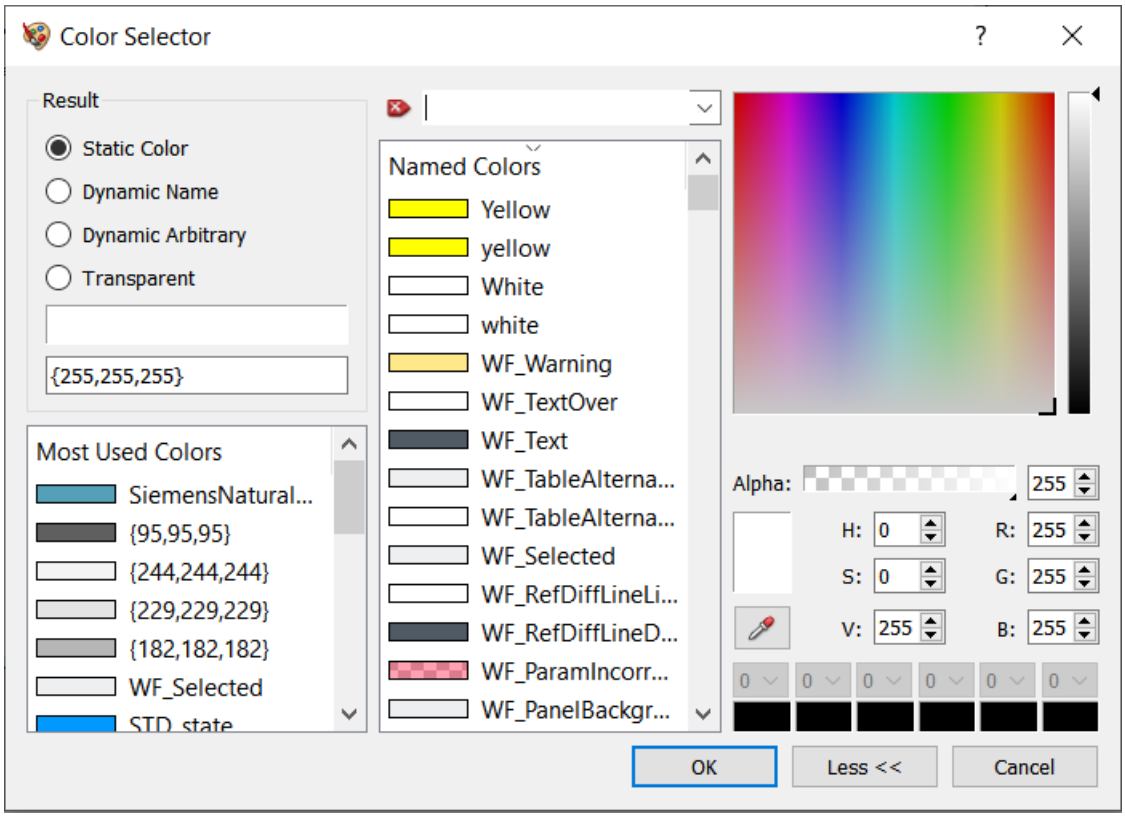


В верхней части редактора gedi есть графические примитивы, выбирите и нарисуйте на панели прямоугольник.

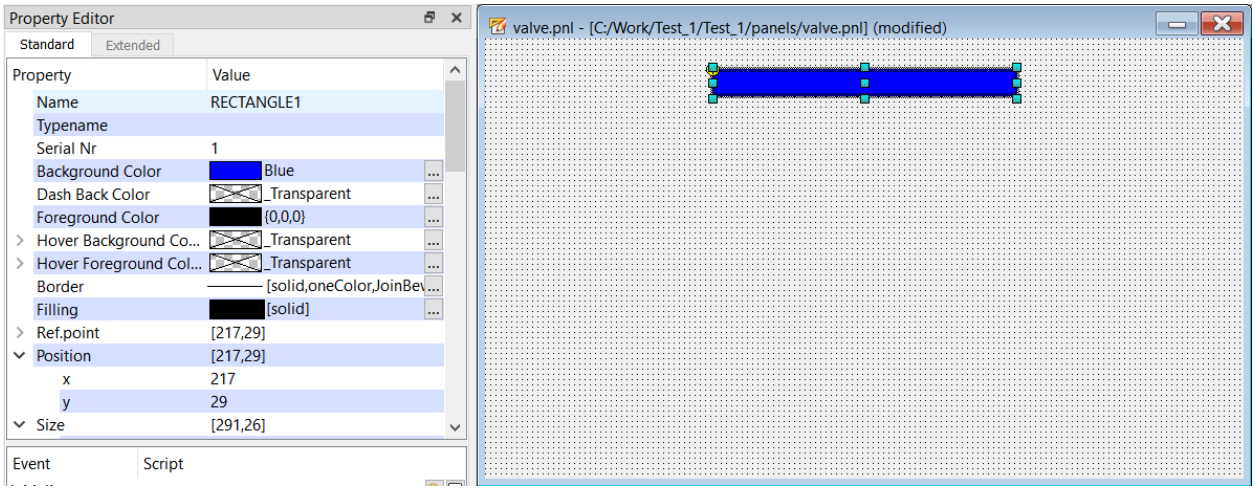
Обратите внимание на окно свойств и событий графического объекта, находящееся слева от самой панели, оно нам понадобится. К примеру, для изменения цвета фона объекта необходимо найти свойство «Background color».



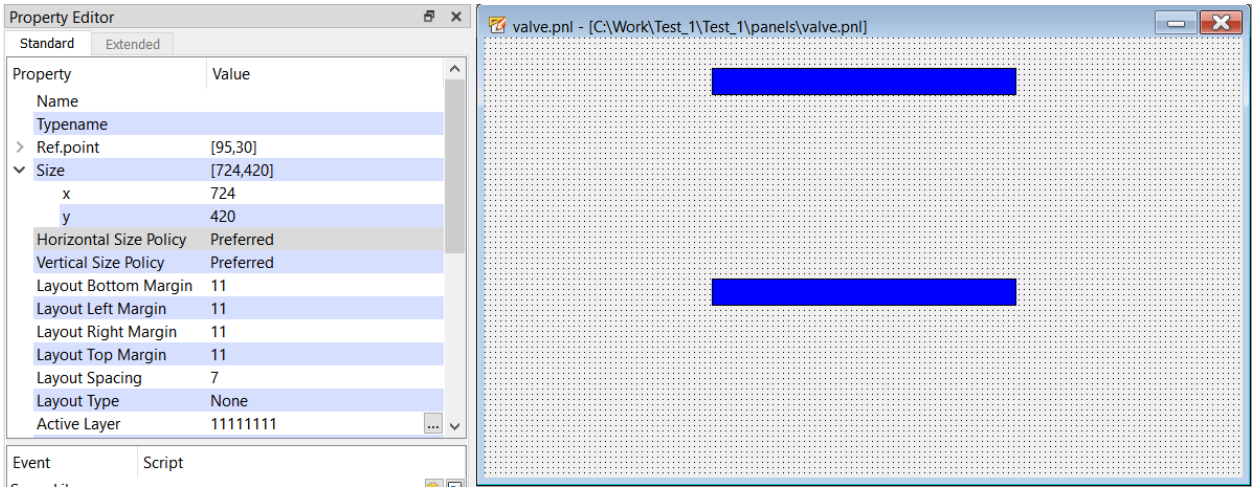
При двойном клике на этом свойстве появляется таблица цветов.



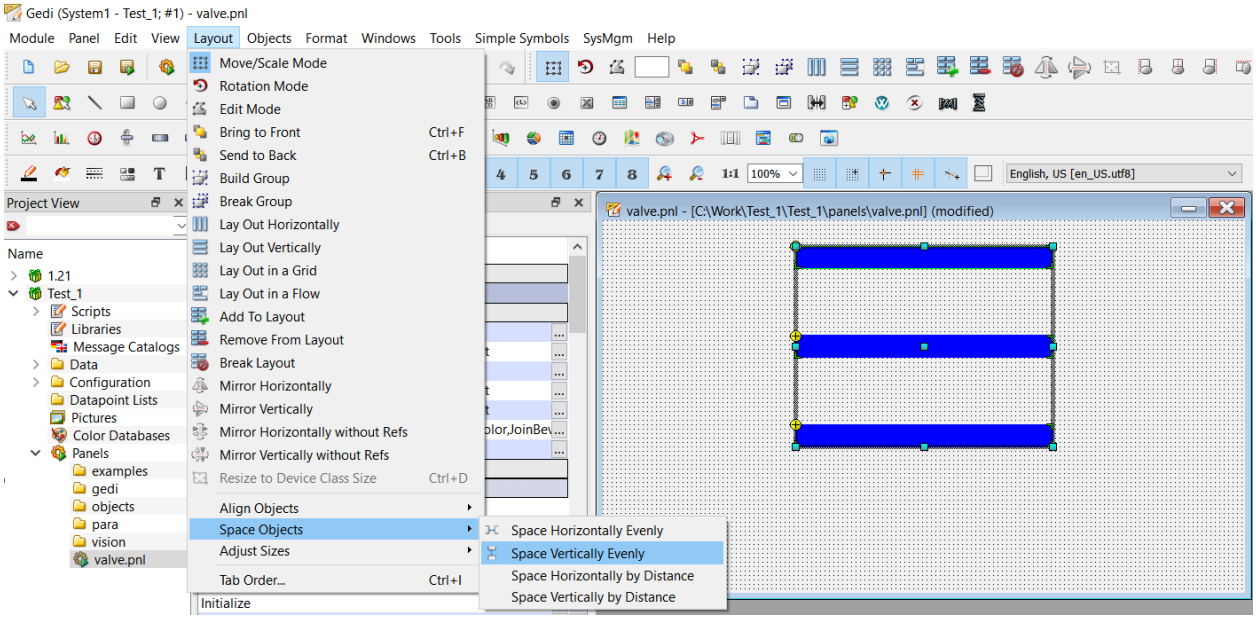
Система позволяет не только выбрать цвет из готового списка. Для создания своего цвета необходимо открыть палитру цветов, задать цвет и его имя, после чего он появится в проекте. Для того чтобы задать цвет RGB, необходимо в окне ниже нажать кнопку More, вбить цвет числами или выбрать его мышью (справа) и убедиться, что он меняется в левой части экрана.



Сделайте копию фигуры как на скрине ниже.



Сделайте ещё одну копию фигуры. Два крайних прямоугольника (низ и верх) изображают стенки трубы. Центральный прямоугольник – это сама задвижка. Для равномерного расположение элементов друг относительно друга, воспользуйтесь инструментом выравнивания как на рисунке.



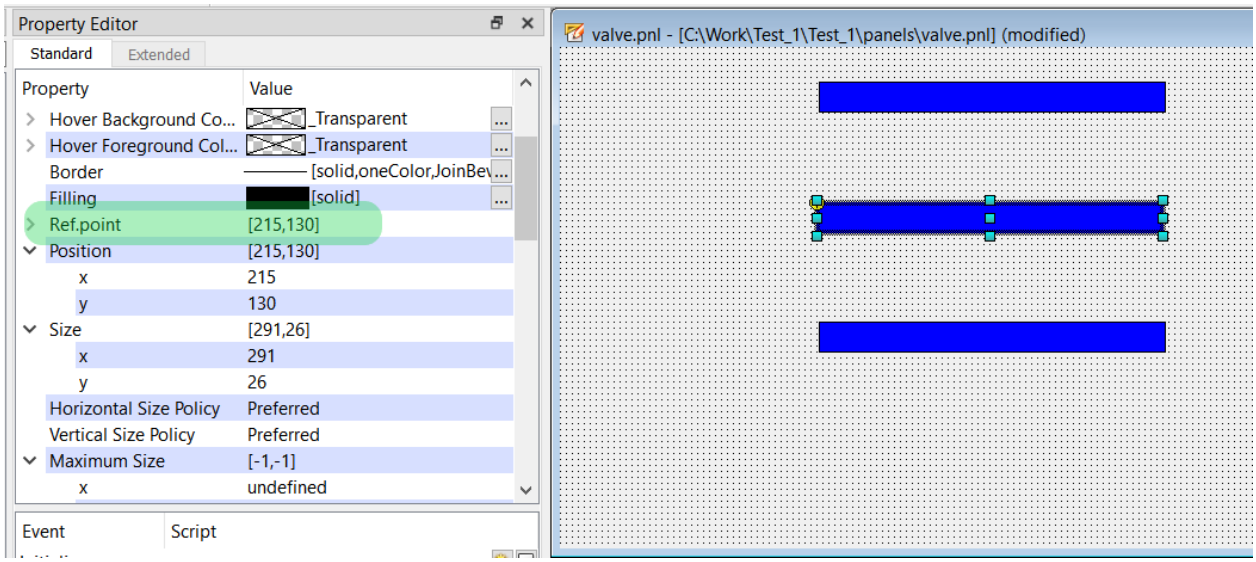
Будем считать условно (как правило всегда) задвижка в нормальном состоянии (без оказанного на него воздействия) находится в закрытом состоянии. Для этого тела заслонки на мнемосхеме необходимо повернуть на 90`.

Каждый объект и каждая панель в редакторе имеет опорную точку (жёлтый кружок). При создании точка обычно размещается там, где вы начинаете рисовать объект. Исключением здесь является круг/эллипс (опорная точка в центре).

Опорная точка панели может быть скрыта. Строка меню “View → Show Panel Reference Point Ctrl + R”.

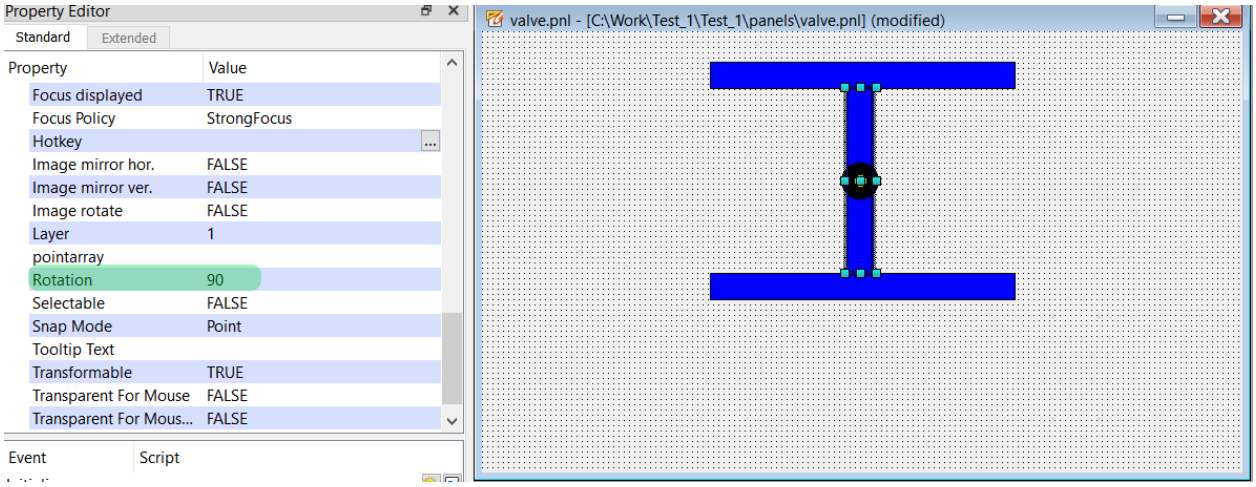
Если в свойствах объекта задать поворот на 90’, то фигура повернётся относительно опорной точки. Поскольку заслонка задвижки вращается относительно центральной оси, необходимо перенести опорную точку в центр фигуры.

Сдвигать опорную точку необходимо в окне свойств объекта, непосредственно мышью перемещать эту точку нельзя.

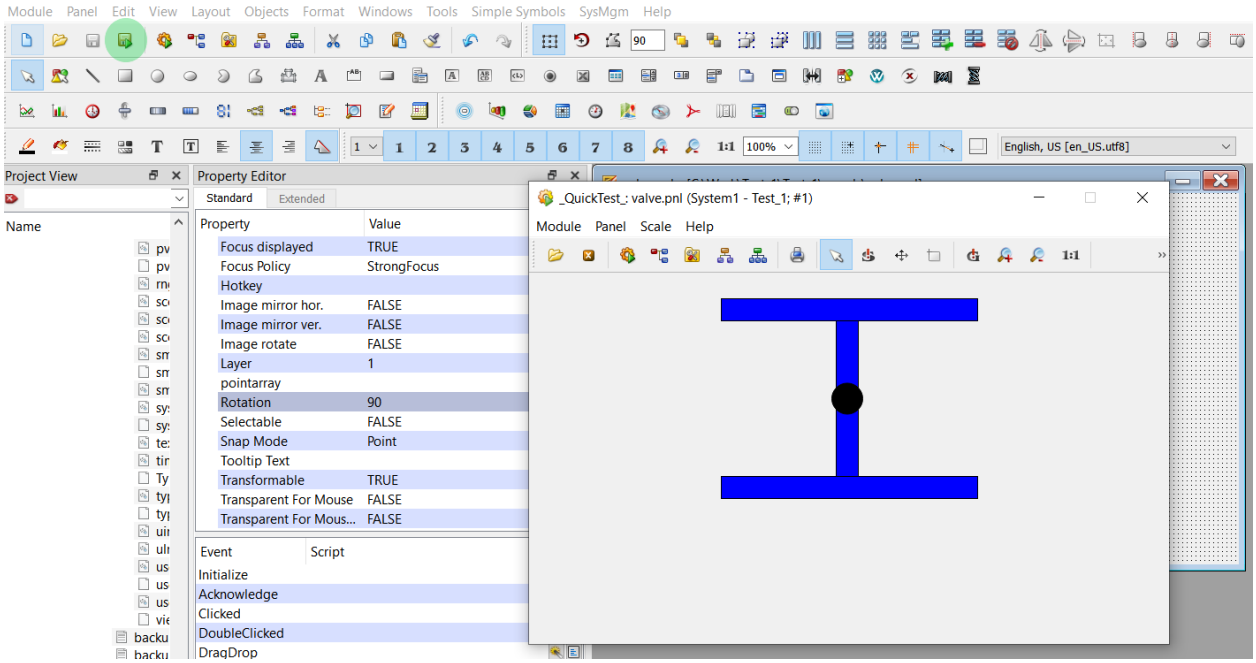


Для удобства нарисуйте окружность и поместите её в центр заслонки, а затем посмотрите на координаты опорной точки у круга и задайте такие же координаты опорной точки заслонки.

Поверните заслонку на 90’, при необходимости скорректируйте размер задвижки (чтобы её края вписывались между стенками трубопровода).



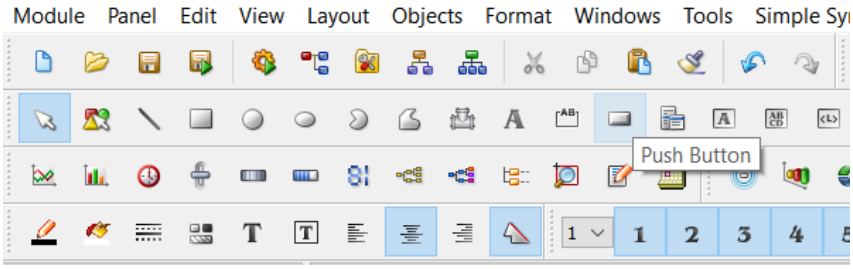
Рядом с кнопкой Save есть кнопка Save and run. Нажмите ее и увидите графическое изображение задвижки в режиме исполнения.



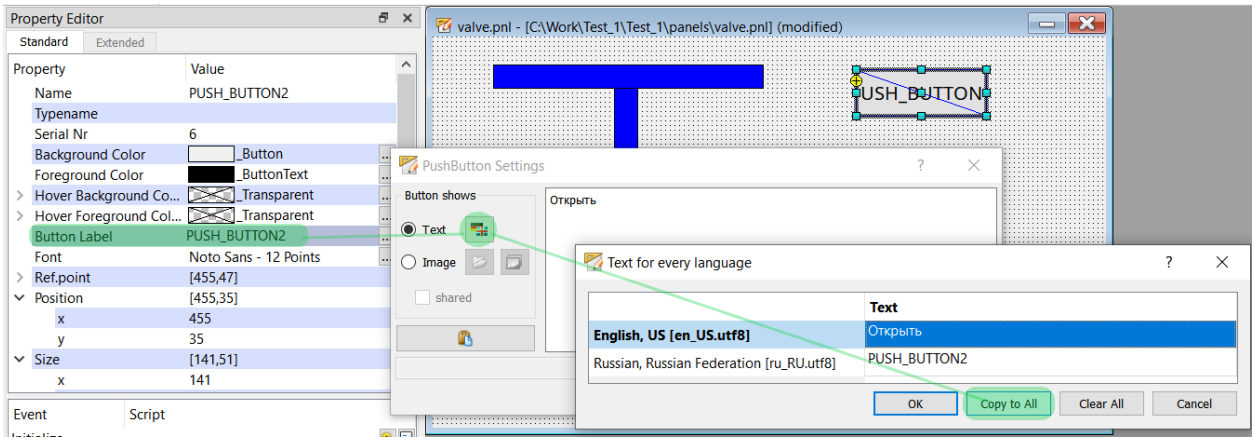
1.5. Динамика

Добавьте на панель кнопки управления задвижкой «Открыть» и «Закрыть». В настоящий момент мы работаем исключительно в APDAR, поэтому реакцию системы на команды реализуем тоже средствами этой системы.

Нарисуйте кнопку при помощи инструмента Push Button и измените название кнопки на «Открыть», отредактировав его свойство «Button label».



Введите сообщение, которое должно быть на кнопке, далее для того, чтобы кнопка выглядела одинаково независимо от того какой язык в системе, вы используете, нажмите на иконку с флагами (см. рисунок), выделите строку с новым именем и нажмите кнопку Copy too all. Далее нажмите кнопку Ok и подтвердите завершение редактирования в следующем окне.



Скопируйте кнопку и переименуйте её в «Закрыть».

Запрограммируйте нажатие кнопки «Открыть». Для этого выделите в редакторе кнопку и в списке событий нажмите на Clicked, вызвав тем самым Script Editor.

Для работы с графическими объектами есть две базовые функции:

getValue – получить значение свойства графического объекта;

setValue – задать свойство графического объекта.

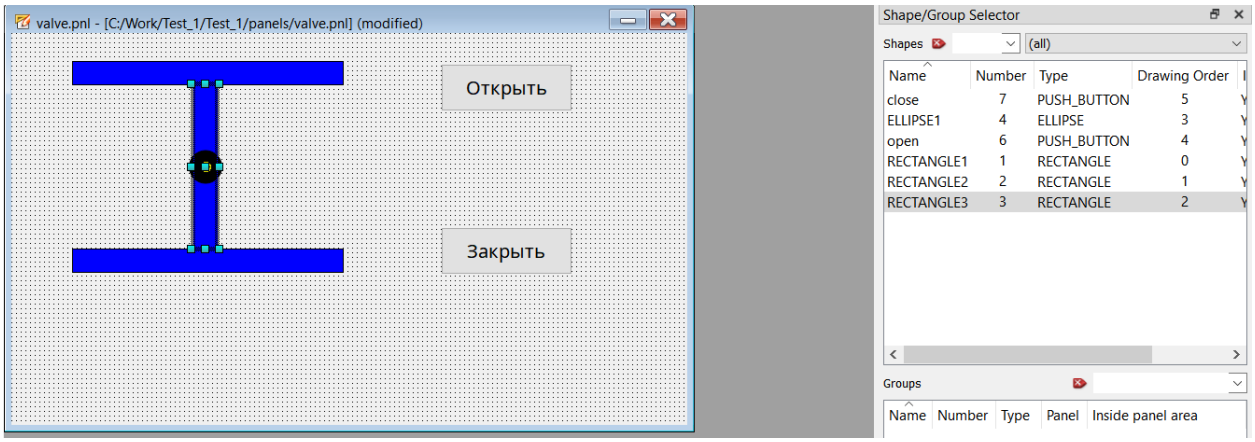
Это высокопроизводительные системные функции. Воспользуемся ими. Для задания уставки угла вращения графическим объектом в теле функции main() напишите следующий текст:

setValue(“ ”, “rotation”, 90);

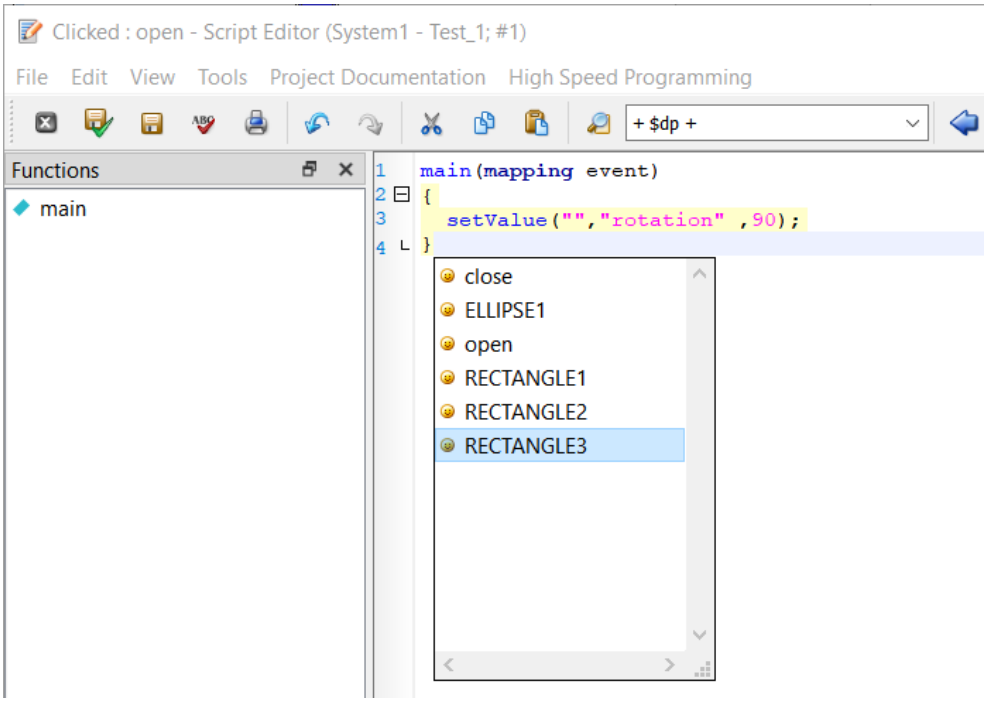
Функция setValue принимает три параметра. Первый это имя графического объекта, второй параметр это свойство, которые необходимо изменить (в нашем случае это rotation) и третий параметр это величина воздействия. Нам нужно повернуть заслонку на 90 градусов.

Чтобы узнать имя объекта (заслонки) выполните следующее:

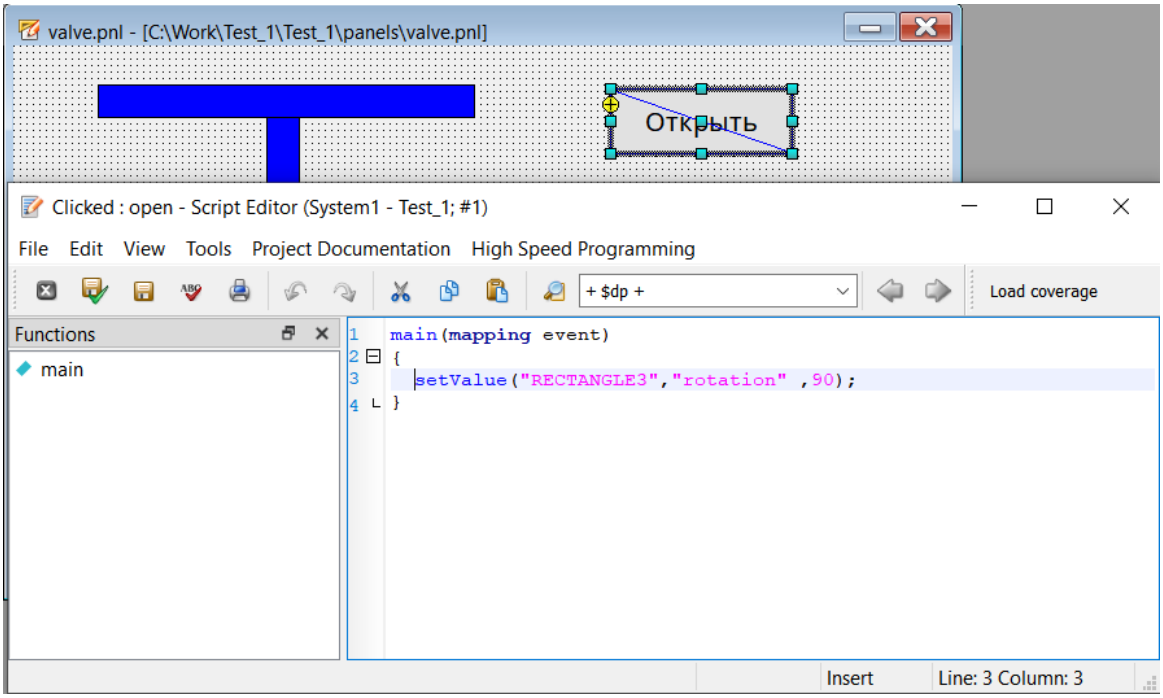
В меню редактора gedi откройте окно перечня графических объектов View → Shape/Group Selector. В открывшемся перечне найдите заслонку. Можно было бы просто выбрать заслонку на панели и прочитать имя объекта в свойствах. Это просто ещё один способ и возможность лучше познакомится с редактором.



Далее, чтобы ускорить ввод либо с целшью не допустить ошибкии в имени при наборе, в редакторе скрипта заходим в пункт меню View → Show Shape List (либо сочетание клавиш Ctrl + O) и выбираем там наш объект.



В итоге текст скрипта кнопки открытия задвижки выглядит следующим образом:

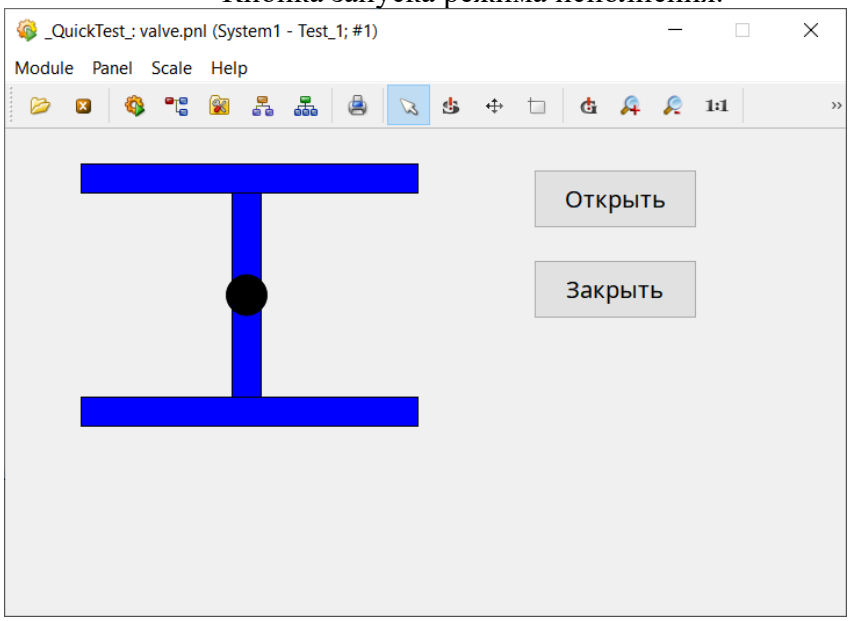


Скопируйте текст скрипта и повторите его для кнопки закрытия, но вместо величины поворота укажите не 90, а 0.

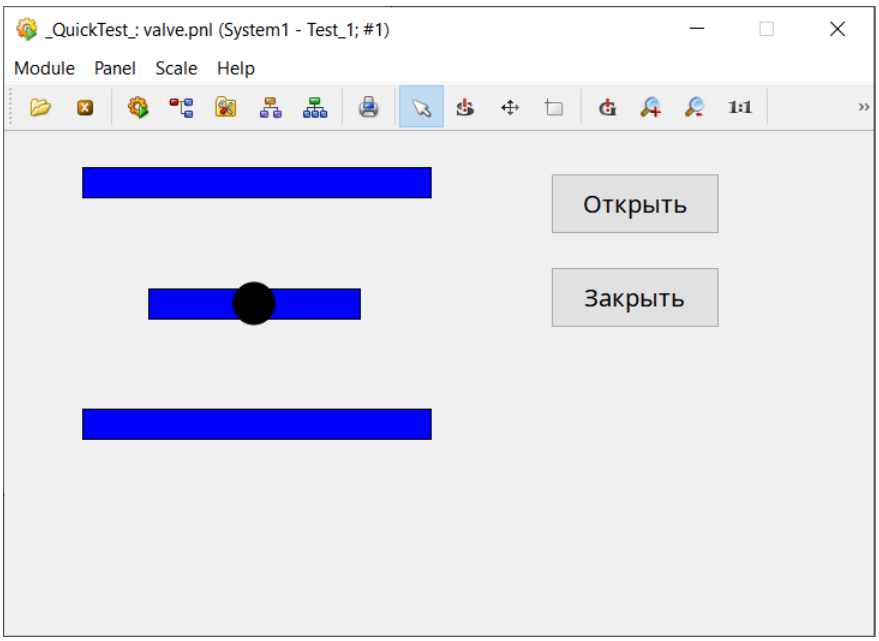
ВНИМАНИЕ. *Редактор скриптов ловит только самые грубые синтаксические ошибки. Ошибки семантики не определяются.*

Далее запустите режим исполнения панели и проверьте работу кнопок управления.

Кнопка запуска режима исполнения.



Нажата кнопка «Закрыть»



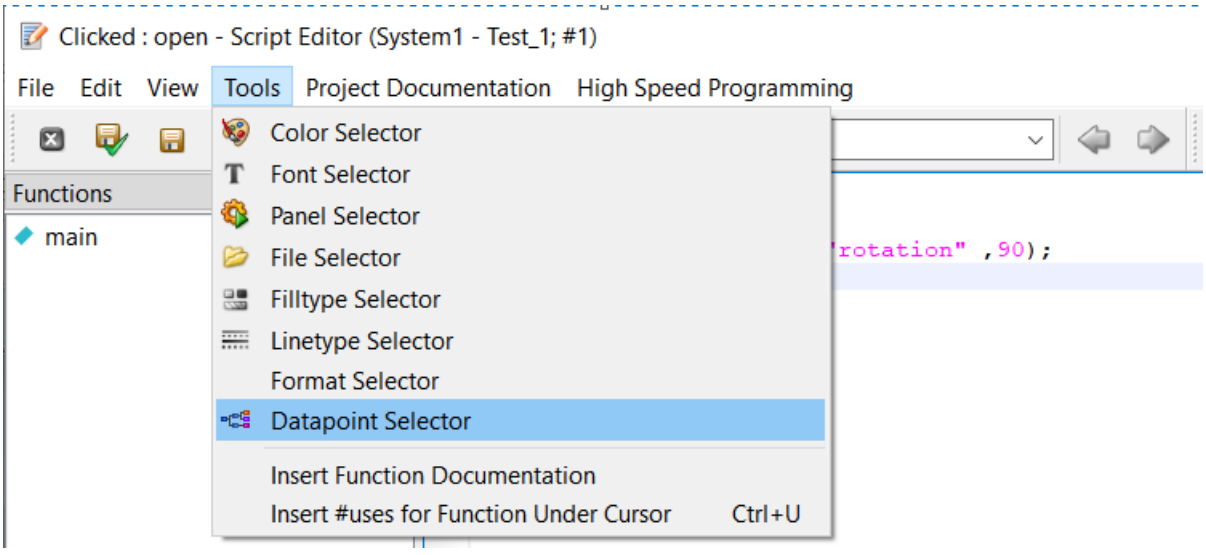
Нажата кнопка «Открыть»

Помимо поворота графического элемента по нажатию кнопок управления также необходимо записывать информацию в точку данных.

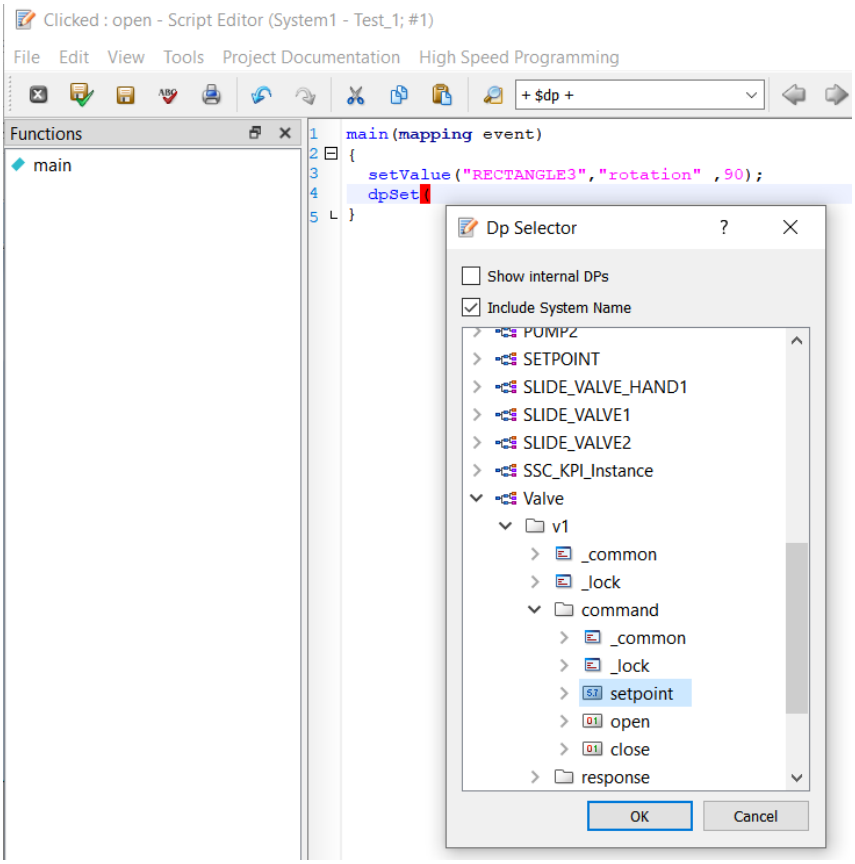
Войдите в режим редактирования скрипта по событию Clicked кнопки «Открыть».

Есть две функции dpSet() – установить значение datapoint элемента и dpGet() – считать значение datapoint элемента.

После функции setValue() добавьте функцию dpSet(). Далее в редакторе скрипта проследуйте до пункта меню Tools → Datapoint Selector.



В открывшемся окне выберете Valve/v1/command/setpoint.



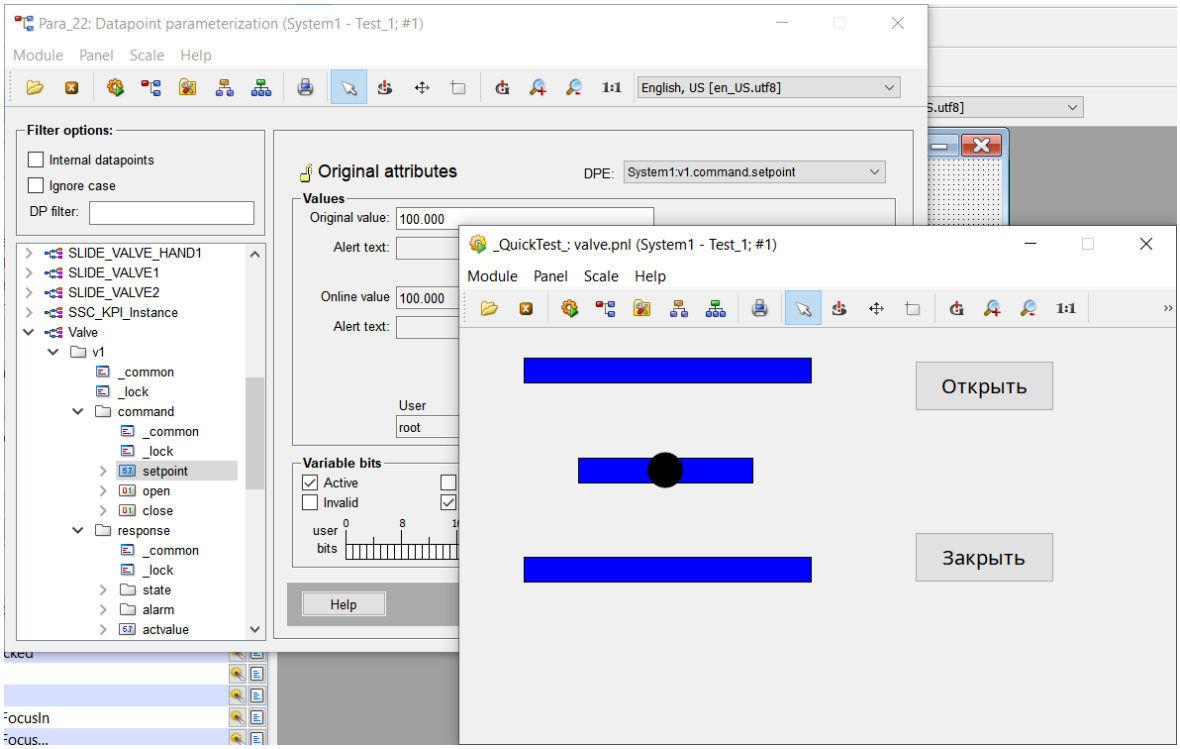
Так как это задание полного открытия задвижки, укажите 100 (%).

```
main(mapping event)
{
  setValue("RECTANGLE3","rotation" ,90);
  dpSet("System1:v1.command.setpoint", 100);
}
```

Скопируйте текст функции и повторите его для кнопки «Закрыть», только вместо 100 укажите 0 (%).

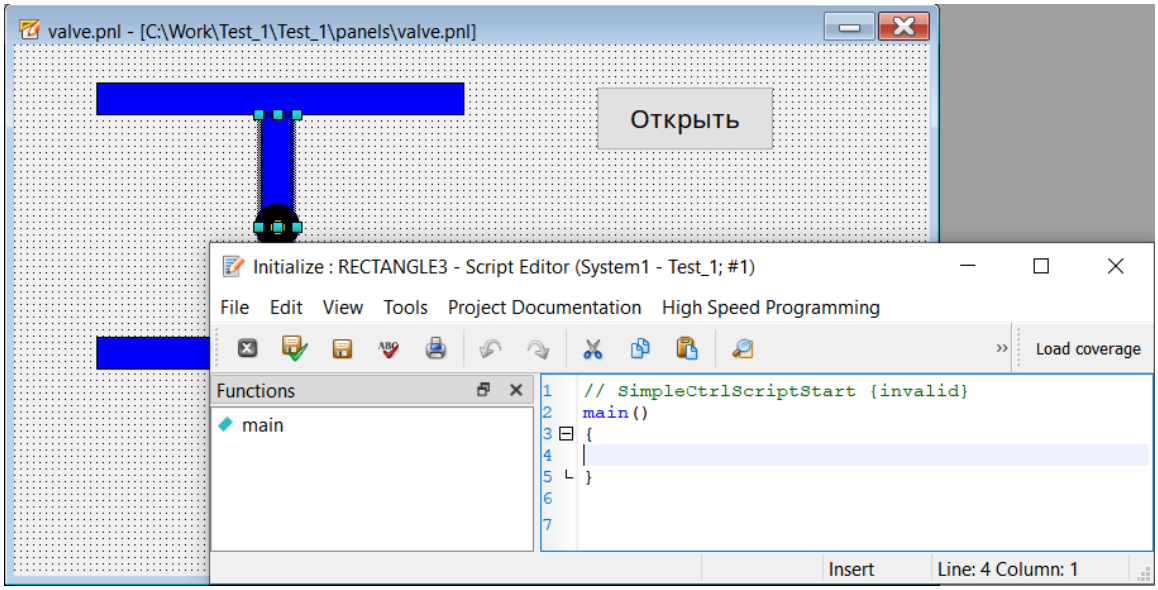
Проверим, меняется ли значение datapoint элемента setpoint.

Запустите редактор para, найдите в нём окно настройки setpoint задвижки v1. Пронаблюдайте изменения значения уставки, подавая команды открыть/закрыть задвижку в режиме исполнения.



Теперь давайте сделаем так, чтобы угловое положение заслонки выставлялось в соответствии с величиной response.actvalue.

В графическом редакторе выделете фигуру поворотной задвижки запустите редактор скрипта для события Initialize.



Необходимо прочитать значение текущего положения, а затем выставить угол поворота заслонки равное этому значению.

Для этого будем использовать две функции dpGet() и setValue().

Объявите переменную типа float, а для функций dpGet() и setValue() укажите параметры как показано ниже:

Main()

{

float value;

dpGet(“System1:v1.response.actvalue”, value);

setValue(“RECTANGLE3”, “rotation”, value);

}

Сохраните и закройте радактор скрипта.

Если запустить панель в режим исполнения и задать в редакторе para новое значение для datapoint элемента v1.response.actvalue, то заслонка повернётся в положение, равное введённому значению.

Теперь задайте новое значение для v1.response.actvalue. Значение в para перезаписалось, но задвижка своего положения не имзенила.

Причина в том, что событие Initialize уже прошло, функция считала значение.

Решение: «Подписка на изменения».

Откройте скрипт события Initialize заслонки для внесения изменений.

Закомментируйте старые строки, добавьте новые:

Main()

{

dpConnect(“rotationCB”, “System1:v1.response.actvalue”);

}

/* функция dpConnect() позваляет подписаться на изменения datapoint элемента. Функция ринимает несколько параметров. Мы будем использовать два параметра. Первый это имя функции, которая будет вызвана. Второй параметр – это имя datapoint элемента, котрый изменился и вызвал функцию. Функция обратного вызова (callback-функция (CB)).

*/

// далее необходимо описать callback-функцию rotationCB

void rotationCB(string dp, float pv) {

/* callback функция принимает два параметра: имя стринговой переменной datapoint элемента, и второй параметр – это его новое значение.

*/

ВНИМАНИЕ: подписка происходит на online значение.

setValue("", "rotation", pv);

// альтернативные варианты записи присвоения:

// this.rotation = pv; как вариант присваения величины вращению.

// RECTANGLE3.rotation = pv; как вариант присваения величины вращению.

}

Сохраните и закройте радактор скрипта.

Если запустить панель в режим исполнения и задать в редакторе para новое значение для datapoint элемента v1.response.actvalue, то заслонка повернётся в положение, равное введённому значению. Теперь заслонка будет поворачиваться на новый угол каждый раз, как actvalue изменится.

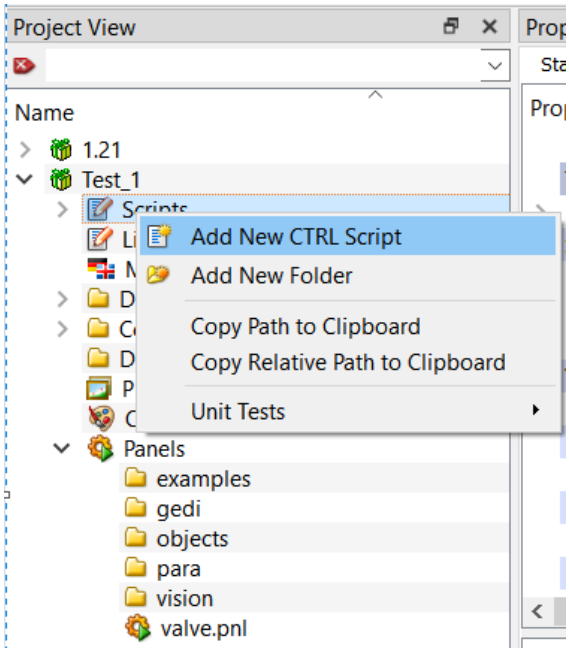
1.6.Глобальные скрипты

В текущем курсе не предусмотрена работа с контроллером, поэтому работу реального оборудования будем имитировать при помощи скрипта.

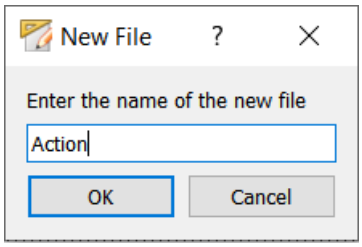
Напишем скрипт, который будет эмулировать поведение задвижки.

Ранее вы задавали значение величины открытия задвижки по нажатию кнопок. В реальной жизни для управления каким-либо агрегатом мы даём команду в виде дискретного либо аналогового сигнала на контроллер, который уже передаёт команду (оказывает воздействие) на исполнительный механизм.

В дереве проекта редактора gedi найдите заголовок Scripts и через контекстное меню выберите Add New CTRL Script.



Дайте скрипту осмысленное имя (в примере он назван Action) и нажмите OK.



Скрипт, с именем, присвоенным вами, будет добавлен в дерево проекта под заголовком Scripts.

Откройте его для редактирования двойным щелчком по имени скрипта.

Скрипт должен вызывать функцию dpConnect, завязанную на одну из переменных команд управления задвижкой.

1. Вызываем dpConnect на изменение значения переменной команды.
2. Создаем callback функцию на изменение значения команды в рамках модели поведения.

Напоминаю, что у функции dpConnect есть два аргумента. Первый аргумент — имя callback-функции (я назвал ее valveSim_CB), второй — имя точки данных, на которую происходит подписка. В итоге функция main скрипта Action выглядит так:

```
main()
{

  dpConnect("valveSim_CB", "System1:v1.command.setpoint");
}
```

Теперь напишем саму callback-функцию. У нее тоже два аргумента — имя точки данных (тип данных string) и «новое» значение этой точки данных (должно соответствовать типу данных «исходной» точки).

Теперь нужно сравнить значение команды и текущее положение. Если команда больше текущего положения, то задвижку нужно открывать. Если команда меньше чем текущее значение, то задвижку нужно закрывать.

Объявим переменную внутри callback функции;

float pv;

Воспользуемся функцией dpGet чтобы запросить значение текущего положения заслонки

dpGet (“System1:v1.response.actvalue”, pv);

Симетируем реальное (постепенное) движение заслонки. Для этого воспользуемся циклом.

While(pv != sp)

{

If(pv > sp)

pv--;

Else pv++;

Необходимо сохранить значение на каждом шаге цикла, чтобы увидеть изменения. Для этого воспользуемся функцией dpSet. Вычисления происходят с высокой скоростью. Для того, чтобы иметь возможность наблюдать движение заслонки необходимо ввести временную задержку.

Для этого воспользуемся функцией delay().

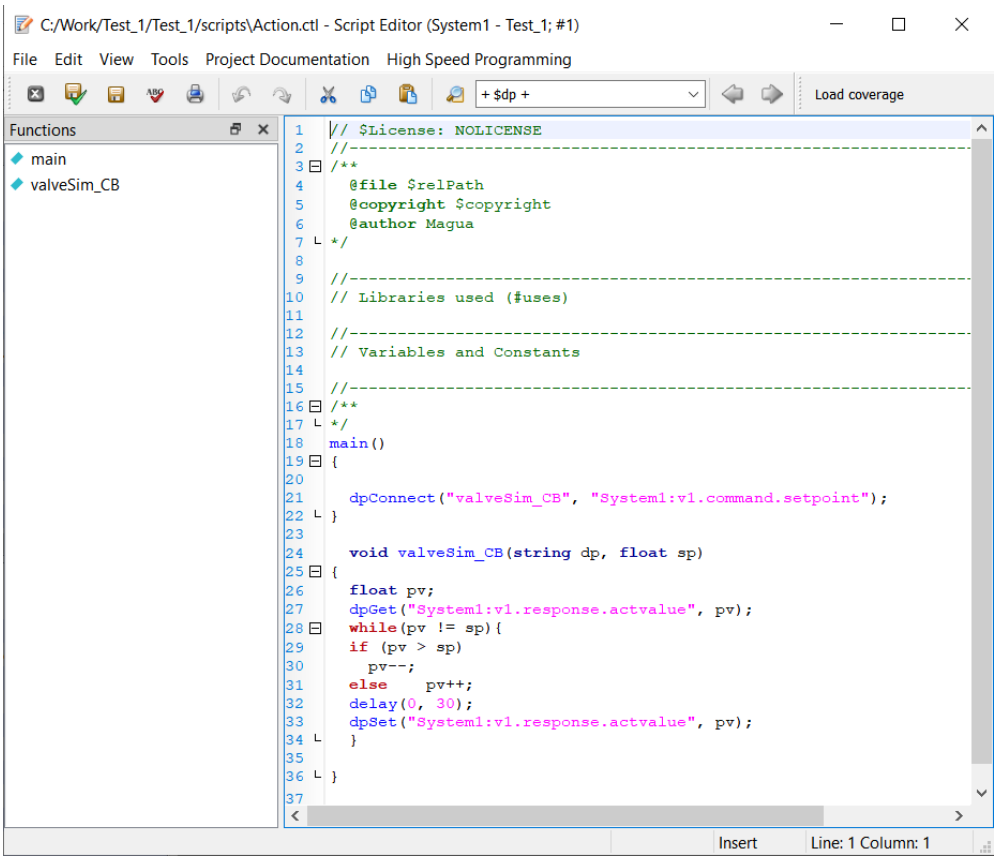
delay(); Функция принимает два параметра. Первый это величина задержки в секундах, вторая - в миллисекундах.

delay(0, 30);

dpSet("System1:v1.response.actvalue", pv);

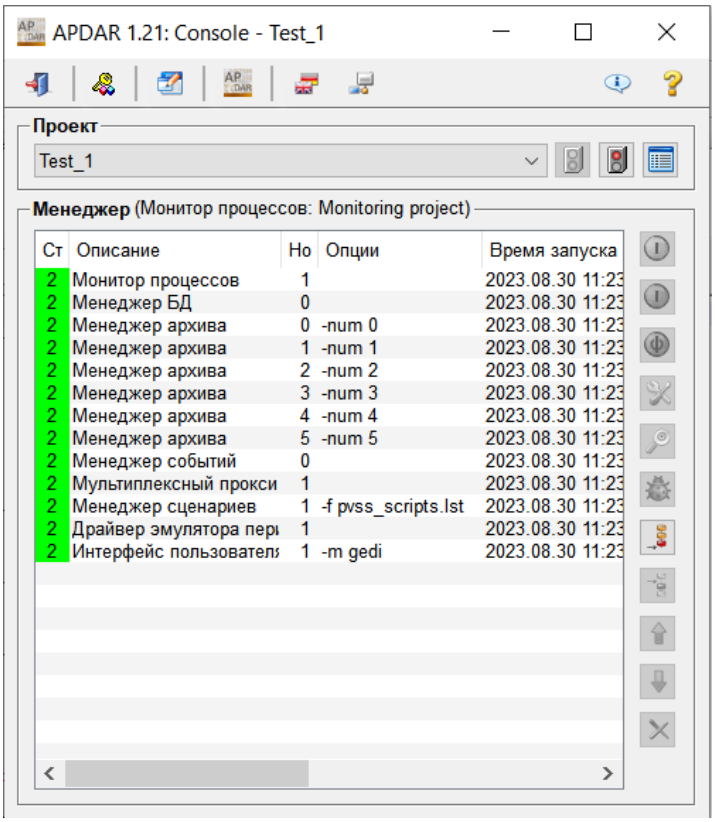
Таким образом от 0 до 100 % задвижка будет открываться за 3000мс, что равно 3 секундам. Для наглядности симмуляции движения задвижки в рамках этого упражнения это приемлемое время. В действительности открытие/закрытие задвижки происходит гораздо дольше.

}



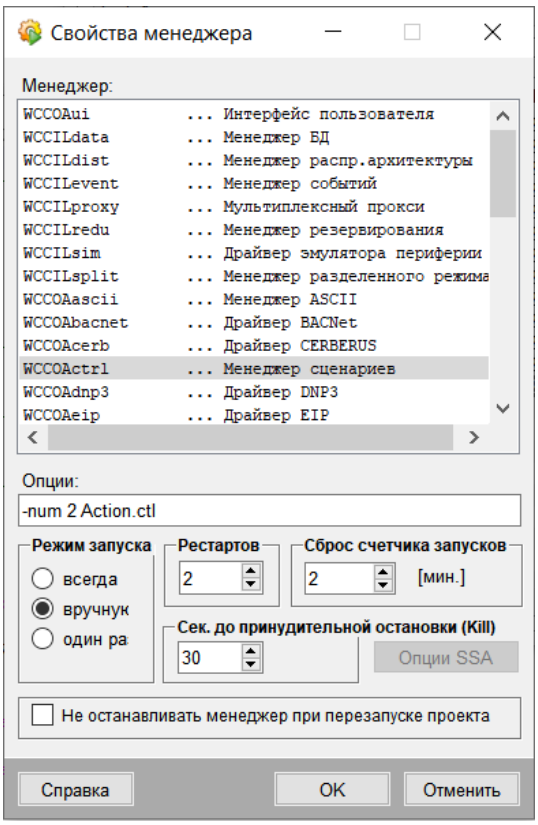
Сохраните и закройте скрипт.

Для исполнения «глобальных» скриптов в APDAR используется Control Manager. Вызов Control Manager необходимо добавить в список менеджеров системы, где в качестве параметра задается имя исполняемого скрипта (Менеджер сценариев). В системе есть уже один вызов Control.



В консоли APDAR нажмите кнопку  Append new manager (Добавить новый менеджер), выбирете в появившемся окне менеджер Control (Менеджер сценариев). Для первого запуска свежесозданного скрипта имеет смысл выбрать режим запуска (Start mode) ручной (manual), чтобы не наблюдать попытки перезапуска в случае ошибок в самом скрипте. В качестве параметров вызова менеджера необходимо указать его номер в системе. В нашем случае это будет номер 2.

Почему 2? Потому что менеджер с номером 1 в системе уже существует. Менеджеры одного типа в системе должны запускаться со своим уникальным номером. Именно одного типа. Это значит, что в системе может быть ui с номером 1 и ctrl с номером 1, а вот двух ui (или ctrl) с одним и тем же номером быть не должно. Поэтому в качестве агрумента запуска укажите строку «-num 2». Кроме того, требуется передать имя исполняемого скрипта. Вызов нового менеджера выглядит следующим образом:

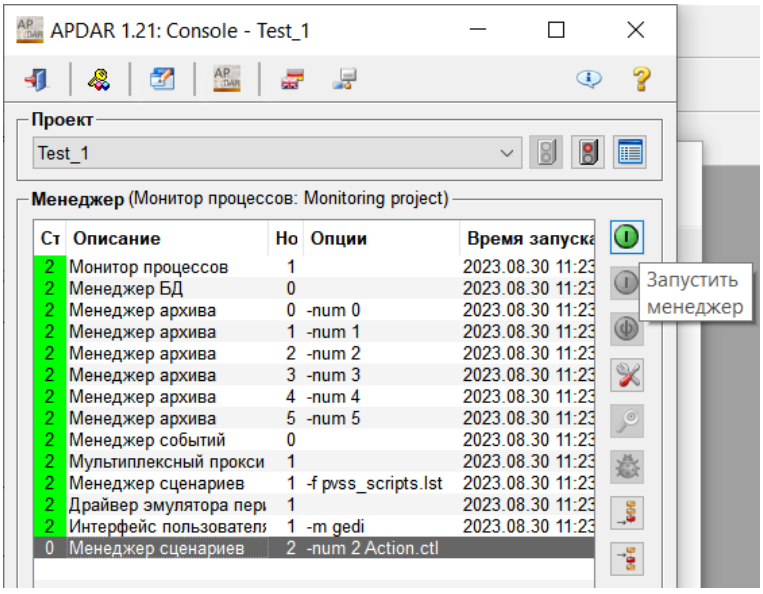


Менеджер скриптов лучше запускать (Start mode/Режим запуска) в автоматичском (always) режиме.

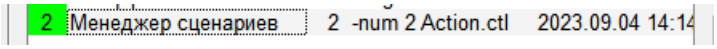
ВНИМАНИЕ. Если ваш созданный ранее скрипт был сохранён не в корневой папке скриптов (как это сделано в примере), а создали новую деректорию (или даже не одну), то в опции менеджера путь до скрипта должен учитывать эти вложения (например NewFolder/Action.ctl).

Завешите добавление нового менеджера, нажав OK.

Дале выполните ручной запуск менеджера. Для этого выделете менеджер в общем списке и нажмите на кнопку Manager Start (Запустить менеджер).



Если всё сделано правильно, после запуска статус менеджера изменится на 2 и окрасится в зелёный цвет.



В режиме исполнения проверьте работу симуляции: реакция положения заслонки по нажатию кнопок.

ВНИМАНИЕ. Control Manager запускает функцию main один раз при старте. После выполнения функции main() необходимые callback-функции продолжают находится в памяти ПК, они исполняются при выполнении условий, указанных в dpConnect (по изменению значения переменной). Если в сам скрипт были внесены изменения, то необходимо вручную остановить соответствующий экземпляр control-менеджера и запустить его заново. Без останова-запуска изменения не вступят в силу.

В режиме исполнения проверьте работу симуляции.

Обаратите внимание, что заслонка ведёт себя странно, подёргивается в начале пути. Это происходит из-за того, что изначально мы задали управление графическим объектом напрямую. В нашем случае это не правильно.

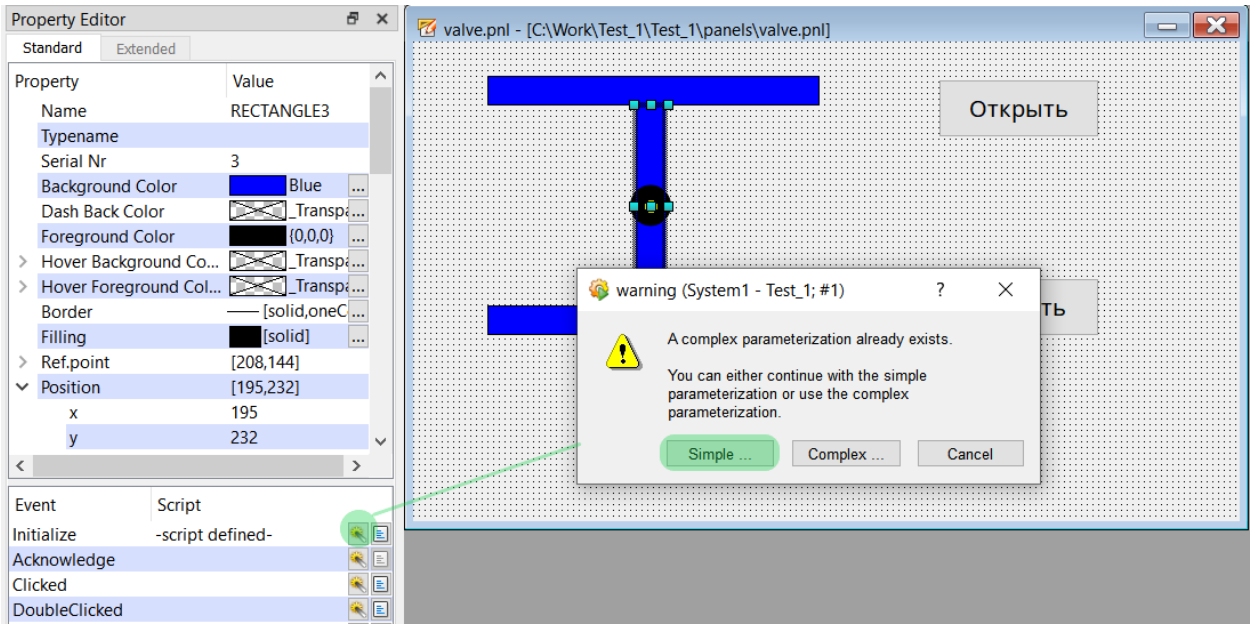
Перейдите в редактор скрипта кнопок управления, закоментируйте строку с функцией setValue.

В режиме исполнения проверьте работу симуляции.

1.7.Линейное масштабирование

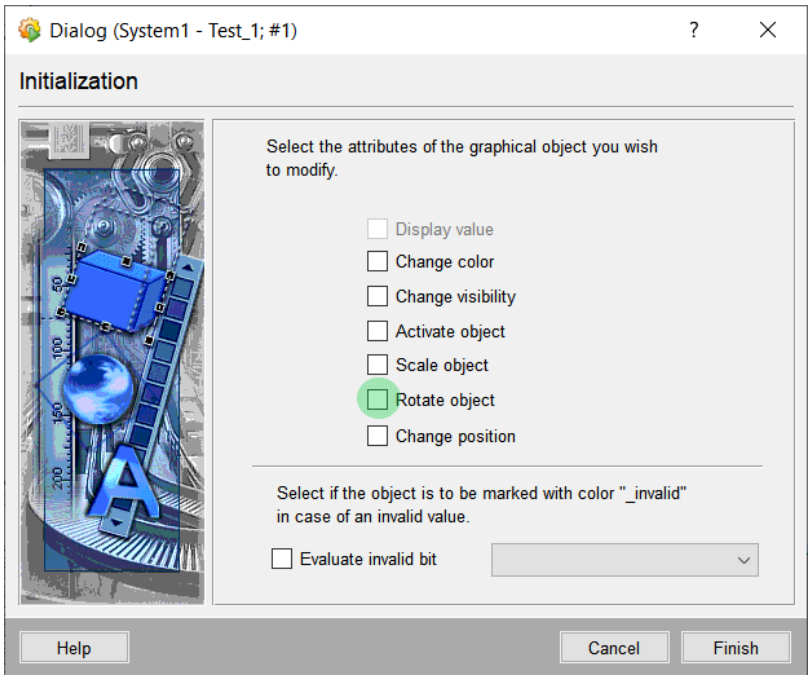
Далее необходимо выполнить линейное масштабирование, чтобы сопоставить уставке открытия задвижки от 0 до 100 % угол поворота от 0 до 90 градусов.

Выделите прямоугольник заслонки и запустите Wizard на событии Initialize.

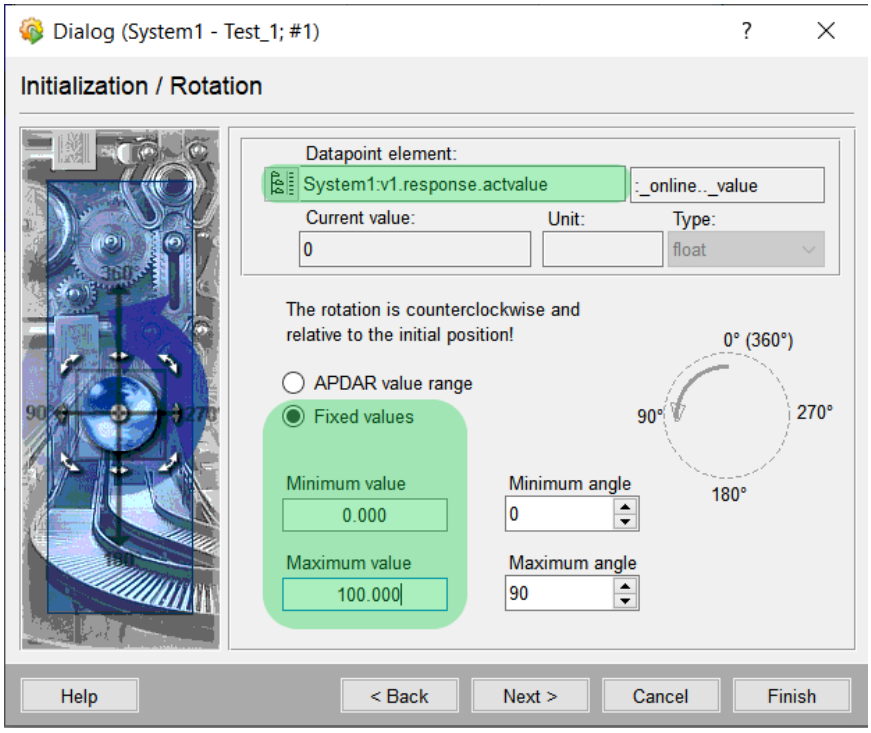


Wizard сообщает, что событие уже запарамтрировано, и спрашивает какое действие выбрать. Вариант Complex переведёт нас к редактированию скрипта вручную, вариант Simle позволит перейти к упрощённому параметрированию при помощи помощника. Выберете Simple.

Выберите Rotate object.



В строке Datapoint element нажмите на кнопку в начале строки и укажите путь до DPE v1.response.actvalue, задайте значения для минимума и максимума (сопоставление величине входного параметра к углу поворота от 0° до 90° а мнемосхеме) 0 и 100 соответственно, нажмите Next. В следующем окне нажмите Finish.



Запустите мнемосхему в режим исполнения, как мы делали ранее. Проверьте угол поворота заслонки при подаче команды «открыть».

Теперь вернитесь к событию «Initialize» для графического примитива «заслонка». Только в этот раз нажмите кнопку Open script editor:

```
// SimpleCtrlScriptStart {invalid}
main()
{
  //float value;
  //dpGet("System1:v1.response.actvalue", value);
 // setValue("RECTANGLE3", "rotation", value);

  dpConnect("rotationCB", "System1:v1.response.actvalue");
  EP_setRotation();
}
void rotationCB(string dp, float pv){
  setValue("", "rotation", pv);
}

void EP_setRotation()
{
  dyn_errClass err;

  if( !dpExists( "System1:v1.response.actvalue:_online.._value") )
  {
    setValue("", "color", "_dpdoesnotexist");
    return;
  }

  dpConnect("EP_setRotationCB",
            "System1:v1.response.actvalue:_online.._value");
  err = getLastError();
  if (dynlen(err) > 0)
    setValue("", "color", "_dpdoesnotexist");

}


void EP_setRotationCB(string dp1, float fNewValue)
{
  float MIN_VALUE = 0;
  float MAX_VALUE = 100;
  float MIN_ROTATION = 0;
  float MAX_ROTATION = 90;

  float fRotation;
  fRotation = ( 1.0 * (MAX_ROTATION - MIN_ROTATION) / (MAX_VALUE - MIN_VALUE)) *
              (fNewValue - MIN_VALUE) + MIN_ROTATION;
  if (fRotation > MAX_ROTATION) fRotation = MAX_ROTATION;
  else if (fRotation < MIN_ROTATION) fRotation = MIN_ROTATION;

  setValue("", "rotation", fRotation);
}

```

Рассмотрим скрипт пристальнее. В первую очередь, присутствует функция main(), это «точка входа» в программу, исполнение всегда начинается с нее.

Wizard (помощник) добавил функцию с именем EP_setrotation. Все функции, которые начинаются с EP_ это функции созданные Wizard-ом.

В первую очередь эта функция выполняет проверку, существует ли в системе точки данных. Если точки данных (точнее, DPE) не существует, то функция прекращает свою работу и выставляет цвет с именем «_dpdoesnotexist» По умолчанию это оттенок

сиреневого. Вы можете отыскать/посмотреть его по имени (_dpdoesnotexist) в редакторе цветов.

Далее идет вызов функции dpConnect().

Функция dpConnect и сам механизм «подписки» — основополагающий механизм системы APDAR.

Что же делает функция dpConnect в данном случае?

dpConnect("EP_setRotationCB", "System1:v1.response.actvalue:_online.._value");

Во-первых, эта функция сообщает системе о наличии Callback-функции EP_setRotationCB (она описана ниже).

Далее, если функция dpConnect() отработала с ошибкой, то вызванная следом функция getLastError() позволяет получить описание ошибки. Если есть описание ошибки, значит длина переменной err отличается от 0, и значит функция dpConnect() отработала с ошибкой, фигура окрашивается в цвет _dpdoesnotexist.

Во-вторых, вызов callback-функции EP_setRotationCB осуществляется по событию System1:v1.response.actvalue:_online.._value, то есть, по изменению элемента точки данных. Ответственным за раздачу сообщений об изменении точки данных у нас, как вы помните, является менеджер событий (EV). Таким образом, изменение угла наклона прямоугольника у нас происходит тогда и только тогда, когда DPE Position точки данных v1 меняет свое значение. Постоянного опроса (polling) данных нет, вся система событийная.

Рассмотрим саму callback-функцию.

```
34  void EP_setRotationCB(string dp1, float fNewValue)
35  {
36    float MIN_VALUE = 0;
37    float MAX_VALUE = 100;
38    float MIN_ROTATION = 0;
39    float MAX_ROTATION = 90;
40
41    float fRotation;
42    fRotation = ( 1.0 * (MAX_ROTATION - MIN_ROTATION) / (MAX_VALUE - MIN_VALUE)) *
43                (fNewValue - MIN_VALUE) + MIN_ROTATION;
44    if (fRotation > MAX_ROTATION) fRotation = MAX_ROTATION;
45    else if (fRotation < MIN_ROTATION) fRotation = MIN_ROTATION;
46
47    setValue("", "rotation", fRotation);
48  }
49
```

Первый параметр этой функции — точка данных (символьное имя точки данных, dp1), генерирующая изменения. Второй параметр — значение этой точки данных (fNewValue). Тип этого параметра зависит от типа DPE, на который происходит подписка.

Кроме проверки значений параметров эта функция содержит вызов setValue. Ее параметры:

1-ый — имя объекта, значение которого мы меняем. В данном случае пустое имя подразумевает текущий объект (наш прямоугольник);

2-ой (rotation) — имя свойства, которое мы меняем, а меняем мы угол наклона, это имя свойства берется из документации и из графического редактора свойств объекта.

3-ий (fRotation) — значение свойства.

Комментарии в конце скрипта так же созданы Wizard-ом. В них хранятся настройки: имя функции, имя datapoint-элемента, на который произведена подписка, атрибуты, константы. Менять и удалять комментарии не нужно.

```
// SimpleCtrlScript {EP_setRotation}
// DP {System1:v1.response.actvalue}
// DPConfig {:_online.._value}
// DPType {float}
// PVSSRange {0}
// Min {0}
// Max {100}
// MinRotation {0}
// MaxRotation {90}
// SimpleCtrlScriptEnd {EP_setRotation}
```

Далее после проверки работы скрипта в режиме исполнения, вновь откройте скрипт прямоугольника (заслонки) и выполните блочное комментирование части скрипта, что мы писали ранее, как это сделано на скрине. Этот код нам уже больше не нужен.

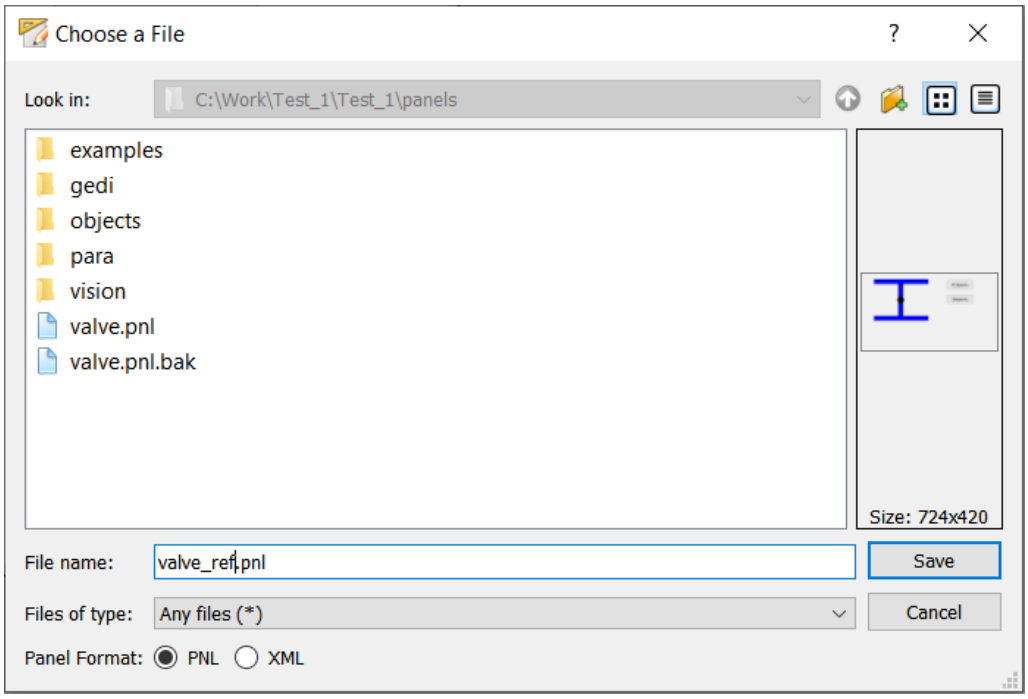
```
7
8     //dpConnect("rotationCB", "System1:v1.response.actvalue");
9     EP_setRotation();
10  }
11  /*
12  void rotationCB(string dp, float pv){
13    setValue("", "rotation", pv);
14  }
15  */
16  void EP_setRotation()
```

1.8. Повторное использование объектов

В реальных проектах типовых исполнительных механизмов и устроств может быть десятки, сотни и тысячи. Рисовать для каждого панель управления и обвязывать её сигналами – это очень долго, трудно и ресурсозатратно. Кроме того, изменения в составе таких панелей либо их визуальный ряд могут вносится часто на этапе ПНР и в процессе эксплуатации. Править каждый экземпляр панели всякий раз - это путь в никуда.

Рассмотрим вариант создания шаблона для панели управления задвижкой, который будет вызываться для всех устроств данного типа.

Сделайте копию нарисованной панели управления, для чего сохраните её под новым именем. Для удобства работы в будущем с панелью в текстовом редакторе, сохраните её с разрешением .xml.



Поскольу вы сделали копию панели пересохранив её под новым именем, далее в редакторе gedi у вас открыта имеена она.

Откройте скрипты панели для редактирования. Для этого проследуйте по пути Edit → Edit Panels Scripts (либо сочетание клавиш Ctrl+E).

```
-// [RECTANGLE3] [3] - [Initialize]
// SimpleCtrlScriptStart {invalid}
main()
{
  //float value;
  //dpGet("System1:v1.response.actvalue", value);
 // setValue("RECTANGLE3", "rotation", value);

  //dpConnect("rotationCB", "System1:v1.response.actvalue");
  EP_setRotation();
}
/*
void rotationCB(string dp, float pv){
  setValue("", "rotation", pv);
}
*/
void EP_setRotation()
{
  dyn_errClass err;

  if( !dpExists( "System1:v1.response.actvalue:_online.._value") )
  {
    setValue("", "color", "_dpdoesnotexist");
    return;
  }

  dpConnect("EP_setRotationCB",
            "System1:v1.response.actvalue:_online.._value");
  err = getLastError();
  if (dynlen(err) > 0)
    setValue("", "color", "_dpdoesnotexist");

}


void EP_setRotationCB(string dp1, float fNewValue)
{
  float MIN_VALUE = 0;
  float MAX_VALUE = 100;
  float MIN_ROTATION = 0;
  float MAX_ROTATION = 90;

  float fRotation;
  fRotation = ( 1.0 * (MAX_ROTATION - MIN_ROTATION) / (MAX_VALUE - MIN_VALUE)) *
              (fNewValue - MIN_VALUE) + MIN_ROTATION;
  if (fRotation > MAX_ROTATION) fRotation = MAX_ROTATION;
  else if (fRotation < MIN_ROTATION) fRotation = MIN_ROTATION;

  setValue("", "rotation", fRotation);
}
```

Открывшееся окно содержит все скрипты всех примитивов на странице (панели).

Для всех задвижек скрипты положения и команды управления одинаковы. Отличаются только имена задвижек, то есть имена точек данных: v1, v2, v3,…

Для создания шаблона в скрипт вводится $-параметр (доллар-параметр или ещё доллар-контейнер). Доллар параметр вводится заместо имени точки данных.

Замените в скрипте все упоминания имени системы и задвижки на $dp +.

```
—// [RECTANGLE3] [3] - [Initialize]
// SimpleCtrlScriptStart {invalid}
main()
{
  //float value;
  //dpGet("System1:v1.response.actvalue", value);
 // setValue("RECTANGLE3", "rotation", value);

  //dpConnect("rotationCB", "System1:v1.response.actvalue");
  EP_setRotation();
}
/*
void rotationCB(string dp, float pv){
  setValue("", "rotation", pv);
}
*/
void EP_setRotation()
{
  dyn_errClass err;

  if( !dpExists( $dp +".response.actvalue:_online.._value") )
  {
    setValue("", "color", "_dpdoesnotexist");
    return;
  }

  dpConnect("EP_setRotationCB",
            $dp +".response.actvalue:_online.._value");
  err = getLastError();
  if (dynlen(err) > 0)
    setValue("", "color", "_dpdoesnotexist");

}


void EP_setRotationCB(string dp1, float fNewValue)
{
  float MIN_VALUE = 0;
  float MAX_VALUE = 100;
  float MIN_ROTATION = 0;
  float MAX_ROTATION = 90;

  float fRotation;
  fRotation = ( 1.0 * (MAX_ROTATION - MIN_ROTATION) / (MAX_VALUE - MIN_VALUE)) *
```

Find: "System1:v1 | Next | Previous | Mark | Case sensitive

Replace: $dp +" | Replace | Replace All | Match whole word

End of text reached. Search continued from beginning. | Wrap Scan

Знак «+» подразумевает объединение строк. Сами строки заключены в кавычки.

Так же сделайте замену в комментариях, созданных Wizard-ом. Это необходимо для корректоной работы Wizard при следующем зампуске.

```
  setValue("", "rotation", fRotation);
}

// SimpleCtrlScript {EP_setRotation}
// DP {$dp.response.actvalue}
// DPConfig {:_online.._value}
// DPType {float}
```

Панель готова, но ещё не пригодна для использования. Если вы сейчас запустите панель в режим исполнения, то увидете, что фигура заслонки окрашена в цвет, говорящий о том, что нет связи между графическим объектом и datapoint-ом. Значение $dp не определено.

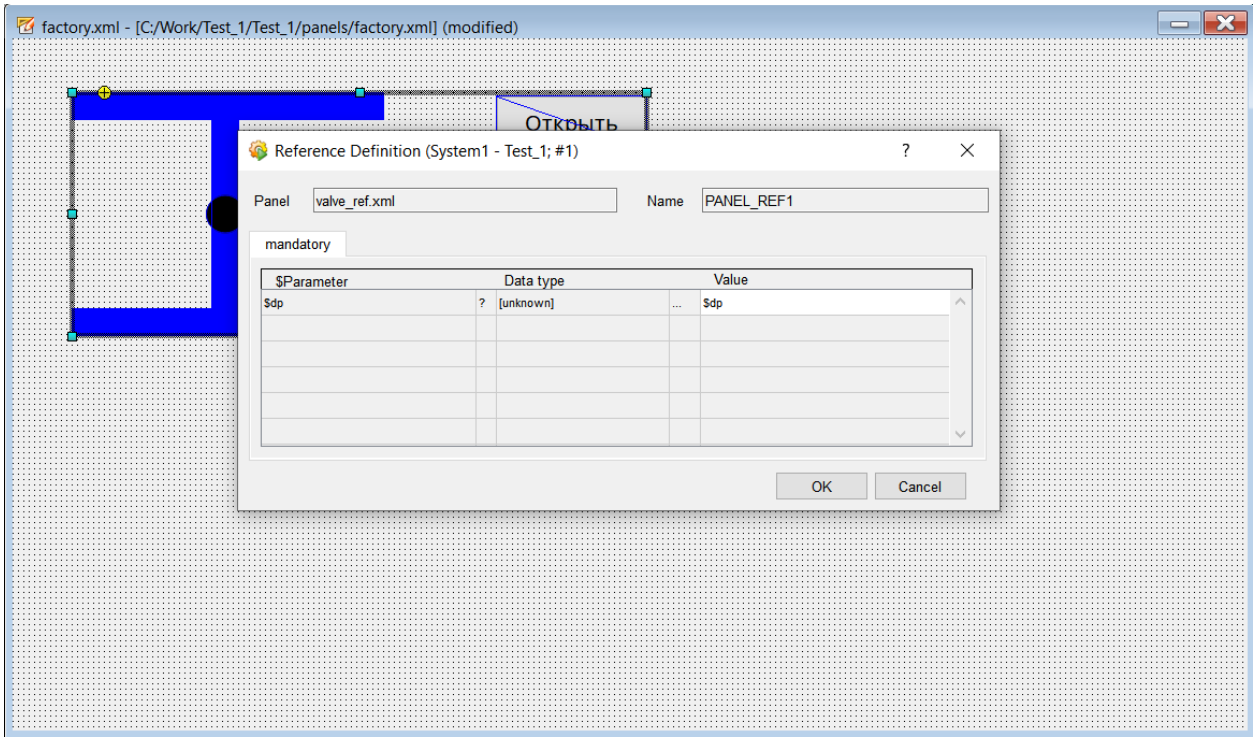
Эту панель теперь нужно использовать не самостоятельно, а в комплексе.

Зададим подстановку $dp для какой-нибудь задвижки при вызоде это панели.

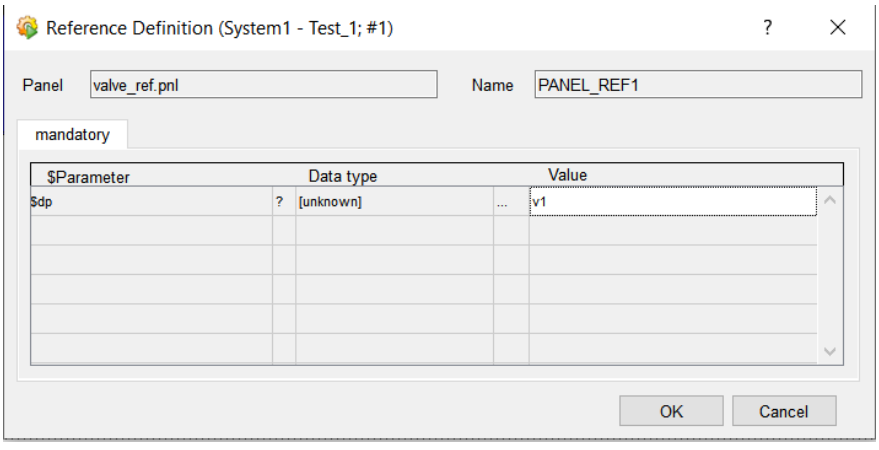
В редакторе gedi создайте новую панель большего размера, чем предыдущая.

Далее из дерева проекта мышью перетащите панель-шаблон в облость новой панели.

Система автоматически зафиксирует присутствие $-праметров и предложит проинициализировать его, заменить формальное значение фактическим.



Задайте имя первой задвижки и нажмите ОК.



Перетащите и разместите на панели ещё два шаблона и проинициализируйте их как задвижки 2 и 3.

Далее запустите панель в рижим исполнения и проверьте работу кнопок и реакцию задвижек на них.

Как вы видите, управляется только первая задвижка. Причина в том, что глобальный скрипт был написан только для первой задвижки. Необходимо внести изменения в скрипт, чтобы он стал универсальным.

Откройте для редактирования глобальный скрипт (Action.ctl).

В тело функции main() необходимо добавить функции dpConnect для всех задвижек.

$-параметр в глобальных скриптах использовать нельзя. Поэтому в функции valveSim_CB выделим из переменной dp общую часть (System1:v) и заменим её на стринговую переменную.

```
main()
{

  dpConnect("valveSim_CB", "System1:v1.command.setpoint");
  dpConnect("valveSim_CB", "System1:v2.command.setpoint");
  dpConnect("valveSim_CB", "System1:v3.command.setpoint");
}

  void valveSim_CB(string dp, float sp)
{
  float pv;
  string dp1;
  dpGet(dp1+".response.actvalue", pv);
  while(pv != sp){
  if (pv > sp)
    pv--;
  else    pv++;
  delay(0, 30);
  dpSet(dp1+".response.actvalue", pv);
  }

}
```

Проинициализируем переменную dp1 при помощи функции dpSubStr, которая возвращает искомую подстроку. В качестве параметра принимает исходную строку. Второй параметр – это целочисленное значение, которое определяет что нужно извчлечь. В данном случае нам подойдёт значение DPSUB_SYS_DP.

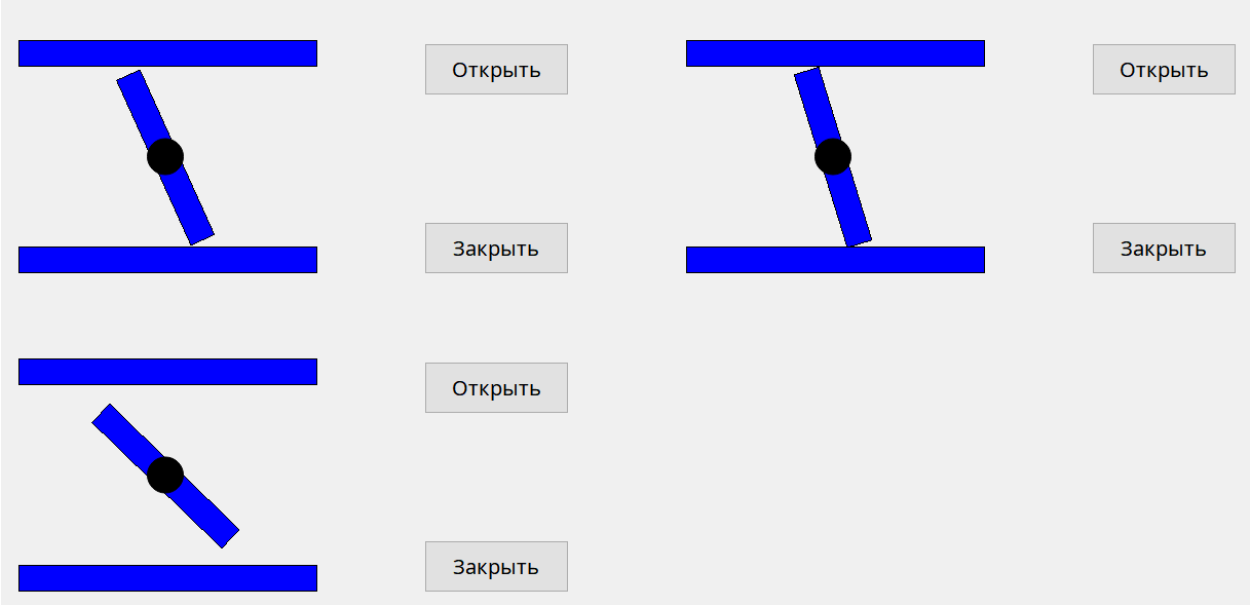
```
main()
{

  dpConnect("valveSim_CB", "System1:v1.command.setpoint");
  dpConnect("valveSim_CB", "System1:v2.command.setpoint");
  dpConnect("valveSim_CB", "System1:v3.command.setpoint");
}

  void valveSim_CB(string dp, float sp)
{
  float pv;
  string dp1;
  dp1=dpSubStr(dp, DPSUB_SYS_DP);
  dpGet(dp1+".response.actvalue", pv);
  while(pv != sp){
  if (pv > sp)
    pv--;
  else    pv++;
  delay(0, 30);
  dpSet(dp1+".response.actvalue", pv);
  }

}
```

Сохраните и закройте редактор скрипта. Перезапустите менеджер сценариев, запускающий данный глобальный скрипт. Запустите панель в режим исполнения и проверьте независимое управление каждой задвижкой.



Усовершенствуем скрипт, что позволит нам подписываться сразу на большое количество точек.

Пересохраните глобальный скрипт под новым именем (Action_v2.ctl).

Представьте ситуацию, что мы не знаем точно какое количество точек данных данного типа есть в проекте, либо их чрезвычайно много и объявлять для каждой из них функцию dpConnect слишком трудозатратно и заёмёт времени.

Вместо того, чтобы использовать отдельный dpConnect() для каждой задвижки, воспользуемся специальной функцией dpQueryConnectSingle() для подключения ко всем задвижкам в базе данных.

dpQueryConnectSingle(<Callback>,<wantanswer>,<userData>,<query>)

Приведите строку к следующему виду, с учётом вашим собственных имён функций:

```
main()
{
  dpQueryConnectSingle("valveSim_CB", FALSE, "", "SELECT '_online._value' FROM '*.command.setpoint' WHERE _DPT=\"Valve\"");
/*
  | dpConnect("valveSim_CB", "System1:v1.command.setpoint");
   dpConnect("valveSim_CB", "System1:v2.command.setpoint");
   dpConnect("valveSim_CB", "System1:v3.command.setpoint");
*/
}
```

Измените callback-функцию по аналогии:

```
   void valveSim_CB(anytype userData, dyn_dyn_anytype res)
//  void valveSim_CB(string dp, float sp)
{
  float pv;
  float sp;
  string dp1;
 // dp1=dpSubStr(dp, DPSUB_SYS_DP);
  dp1=dpSubStr(res[2][1], DPSUB_SYS_DP);
  sp=res[2][2];
  startThread("valveSim_MT", dp1, pv, sp);
}
  void valveSim_MT(string dp1, float pv, float sp)
  {
  dpGet(dp1+".response.actvalue", pv);
  dpSet(dp1+".response.state.run", true); //выставляем бит состояния run
 // если задвижка в пути state.run  = 1
  while(pv != sp){
  if (pv > sp)
    pv--;
  else    pv++;
  delay(0, 30);
  dpSet(dp1+".response.actvalue", pv);
  }
    dpSet(dp1+".response.state.run", false); //сброс бита состояния run

}
```

Обратите внимание, в скрипт так же добавлены строки выставления и сброса бита состояния run задвижки, в случае, если задвижка находится в переходном состоянии, то есть откливается либо закрывается.

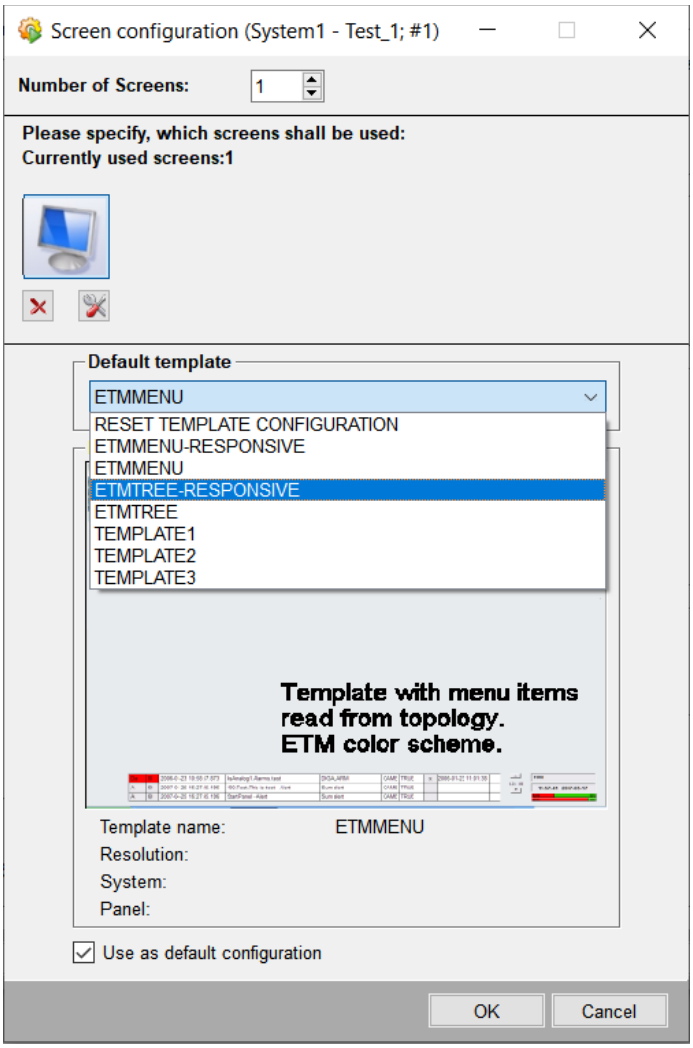
Проверьте работу задвижек в режиме исполнения.

1.9. Создание графической оболочки для оператора

Для создания оболочки в APDAR предусмотрены встроенные шаблоны. На панели инструментов нажмите на кнопку (топология панелей / panel topology)  . В открывшемся окне вы можете выбрать количество мониторов, которою будет на рабочем мечте оператора. Каждый монитор можно настроить индивидуально, кроме того есть возможность настройки рабочей области индивидуально для какждого пользователя.

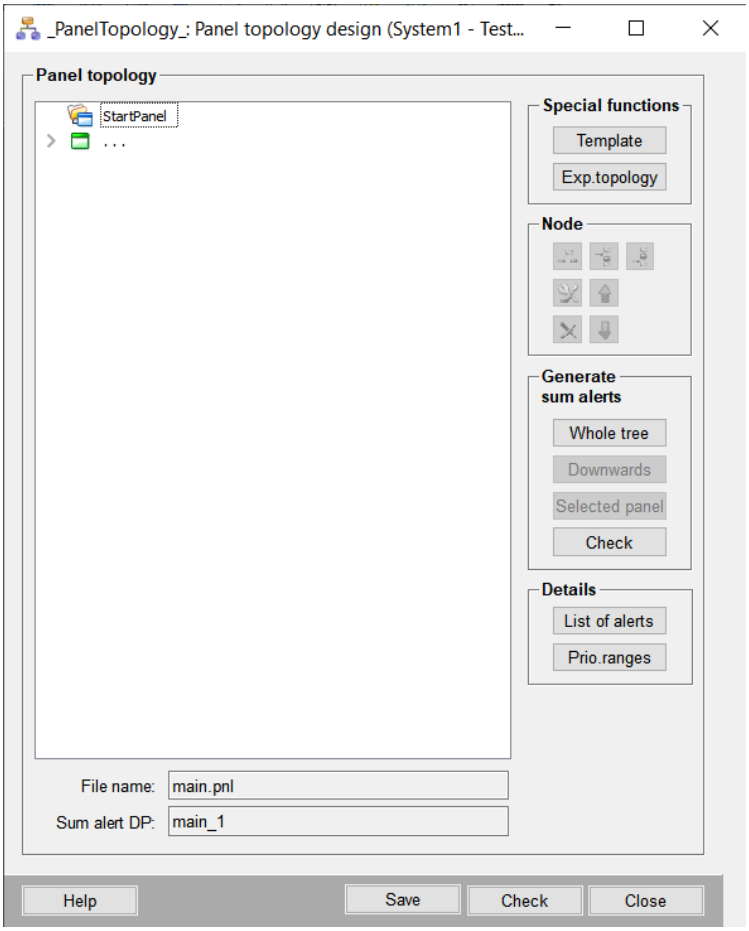
Выберите 1 монитор. Ниже находится раскрывающийся список с заготовленными шаблонами внешнего вида (состава) рабочей области: где будет располагаться дерево проекта(предприятия), где строки варийных и информационный сообщений, где прочая

информация (польщователь, время). Выбирим вариант под именем TEMPLATE1. Нажмите ОК. На вопрос сделать шаблон общим для всех пользователей так же ответьте утвердительно ОК.



Вы попадаете в редактор топологии панелей.

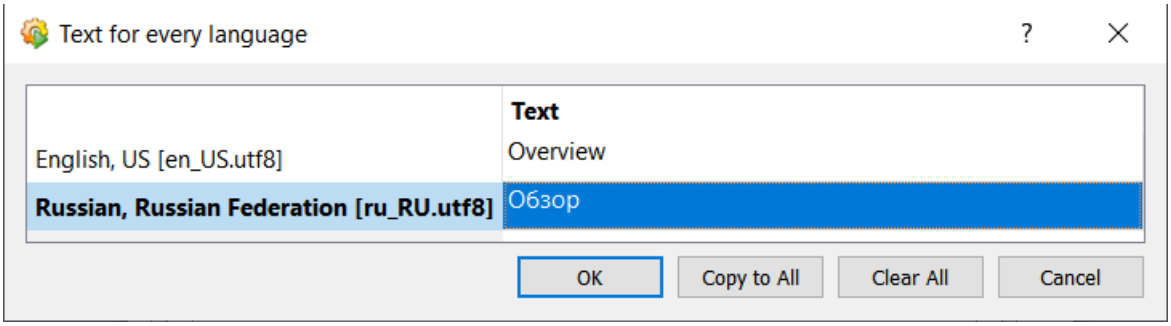
Автоматически создаётся панель с именем main.pnl. Это корневая панель, start panel.



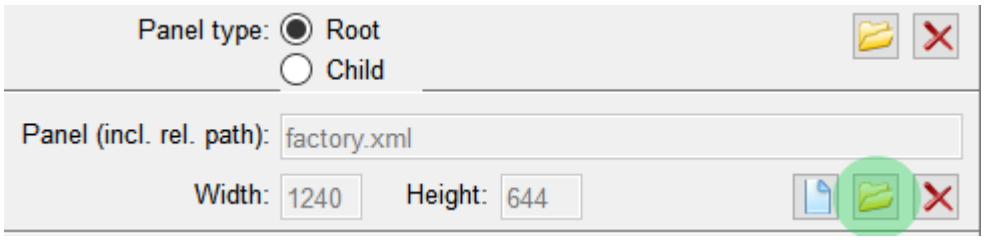
Выполним настройку дерева.

Выделите StartPanel и нажмите кнопку  чтобы добавить новую панель (дочерний узел).

В открывшемся окне задайте имя панели, нажмите ОК.



Далее укажите панель, которая является обзорной (та панель на которой расположены все задвижки).

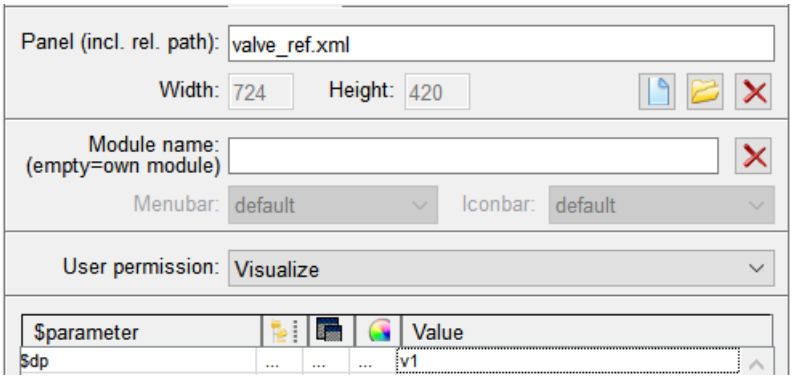


Обратите внимание, тут же есть опция типа панели: Root или Child. Root- это корневые панели, которые занимают всю площадь экрана, child – это всплывающие окна.

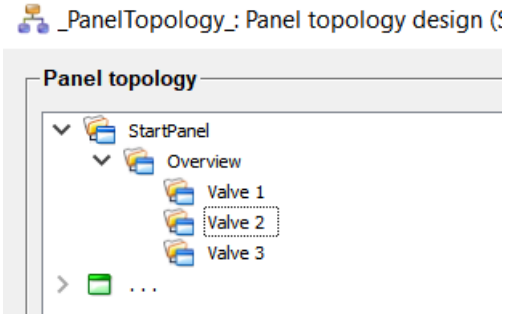
Далее в дереве топологии выделите панель Overview и добавьте дочерний узел уже для неё.

Назовите панель «Valve 1 / Задвижка 1».

В качестве панели укажите панель-шаблон. В таблице окна снизу появится $-параметр. Проинициализируйте его как v1 (имя вашего DP для первой задвижки).



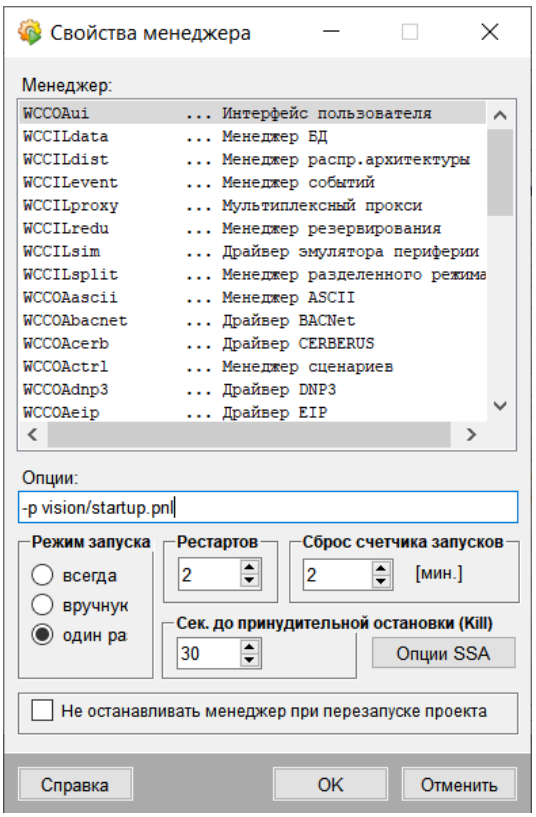
Добавьте дочерние узлы для панели Overview с остальными задвижками. В результате должна получиться такая структура:



Топология рабочего места создана. Теперь необходимо добавить менеджер для её запуска.

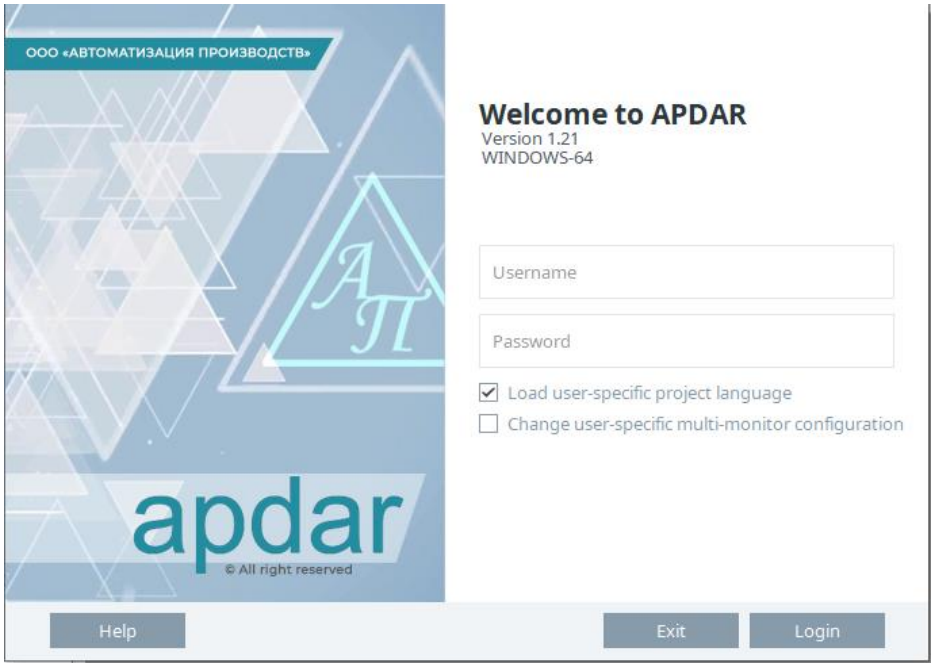
Откройте консоль и добавьте новый менеджер типа User Interface (Интерфейс пользователя) с такими опциями запуска:

-p vision/startup.pnl (ключ p значит panel)

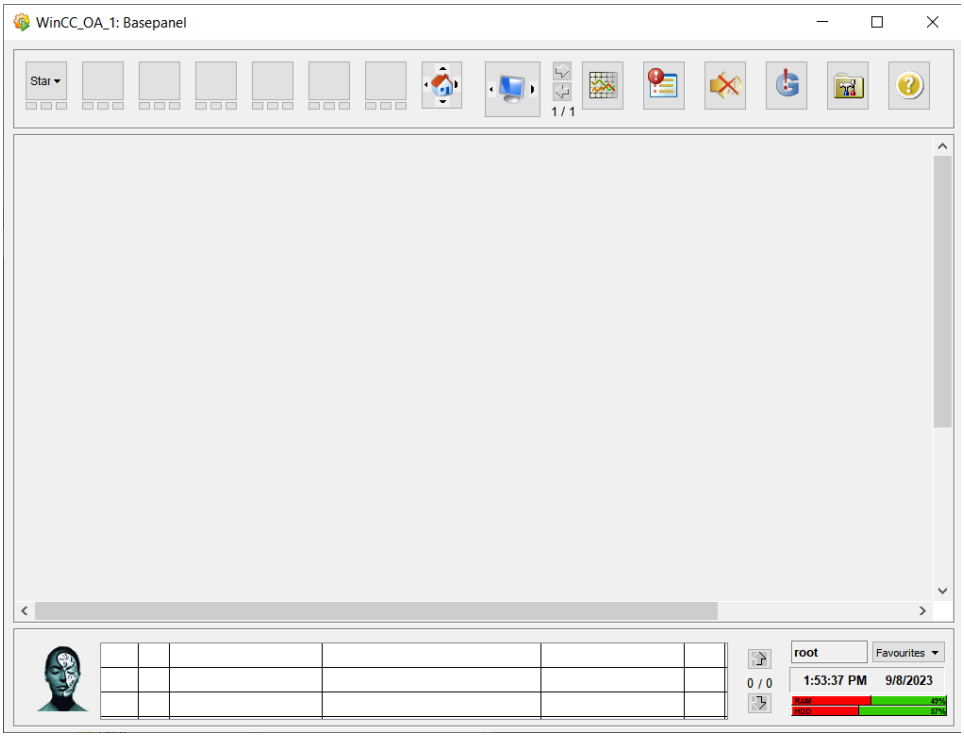


Завершите процесс добавления менеджера, нажав ОК.

Запустите менеджер в работу. Появится стандартное диалоговое окно входа в систему.



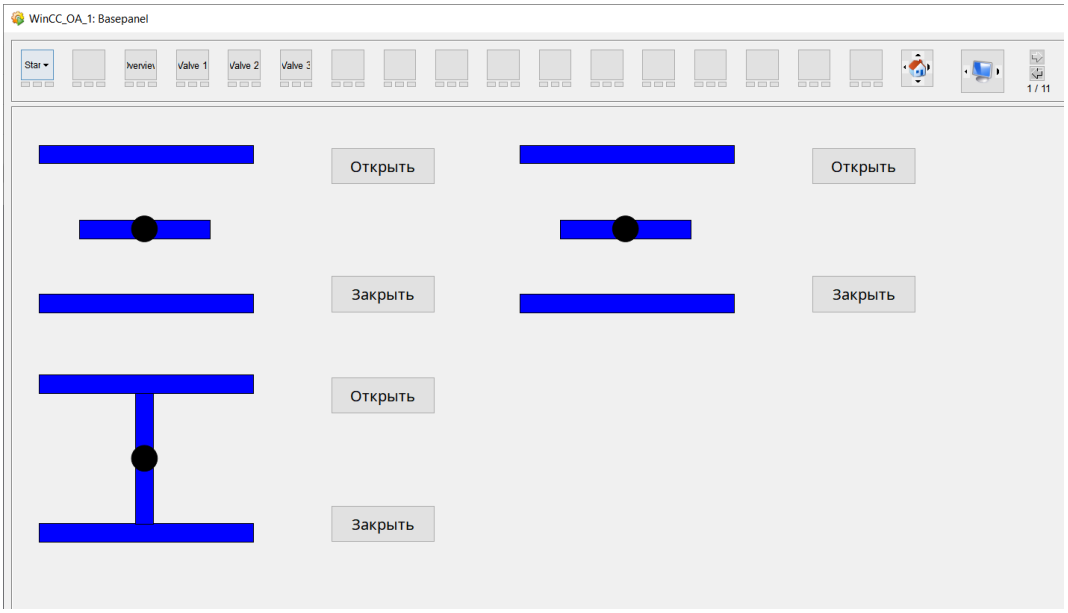
Введите имя пользователя root, пароль отсутствует, нажмите Login. Откроется стандартное окно Overviewиз шаблона.



Окно в зависимости от вашего монитора может быть разного размера и разрешения, которое было выбрано автоматически. Зайдите в настройки топологии панелей, в правом верхнем углу перейди в настройки Template и измените разрешение наваше. Если в выпадающем списке вы не нашли «своего» разрешения, укажите наиболее подходящее вам. Точное разрешение рабочего места оператора можно настраивать если создавать шаблон самостоятельно.

В верхней части окна расположена панель навигации по проекту. В настоящий момент кнопки свободны. Через контекстное меня Properties, вызванное правой кнопкой мыши по кнопке назначьте отдельную кнопку для каждой из ваших панелей из топологии (Overview, valve 1, …).

Фрагмент окна с открытой обзорной мнемосхемой:



1.10 Сообщения

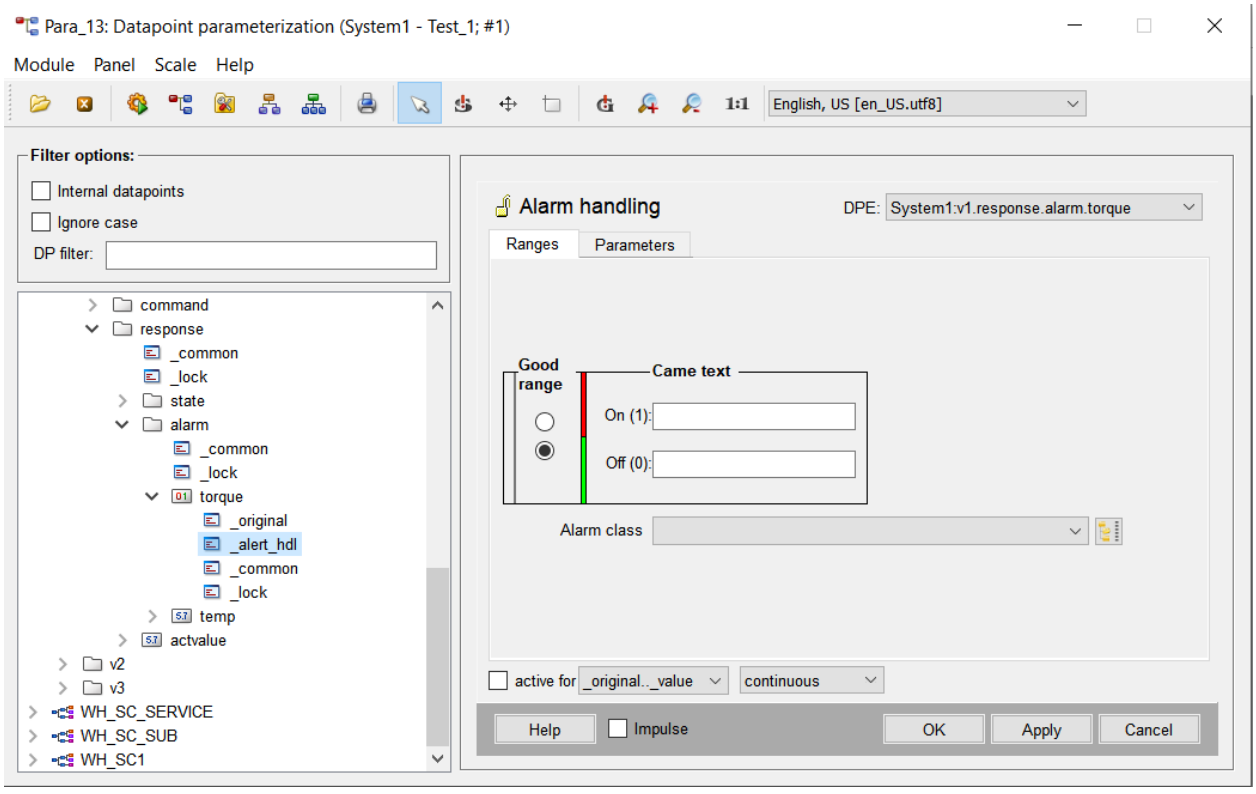
Сообщение – это уведомления пользователя (оператора) системой о событии. События могут быть разными: информационные, системные, предупредительные, аварийные и т.д.

Каждое сообщение включает в себя идентификатор сообщения, текст сообщения и метка времени (время, когда случилось событие) и др.

При формированиее дерева точки данных задвижка, мы добавили битовый элемент в аварии под именем torque (момент). Для того, чтобы изменения этой переменной фиксировались в подсистеме сообщений, необходимо прикрепить на неё соответствующий конфиг.

Откройте редактор para, найдите в дереве проекта v1 → response → alarm и для элемента torque добавьте конфиг под названием alert handling.

Откройте настройки добавленного конфига. Обратите внимание, что внешний вид настройки Alert Handling зависит от типа переменной, к которой относится сам аларм. В настоящий момент мы работаем с типом данных bool, внешний вид окна следующий.



Good range касается значения переменной — какое из значений является «плохим» (т.е. при каком значении считаем, что аларм сработал), а какое — «хорошим» (аларма нет). Здесь же задаем текст аларма.

Задайте этот текст для события при 1, как «came» (пришло).

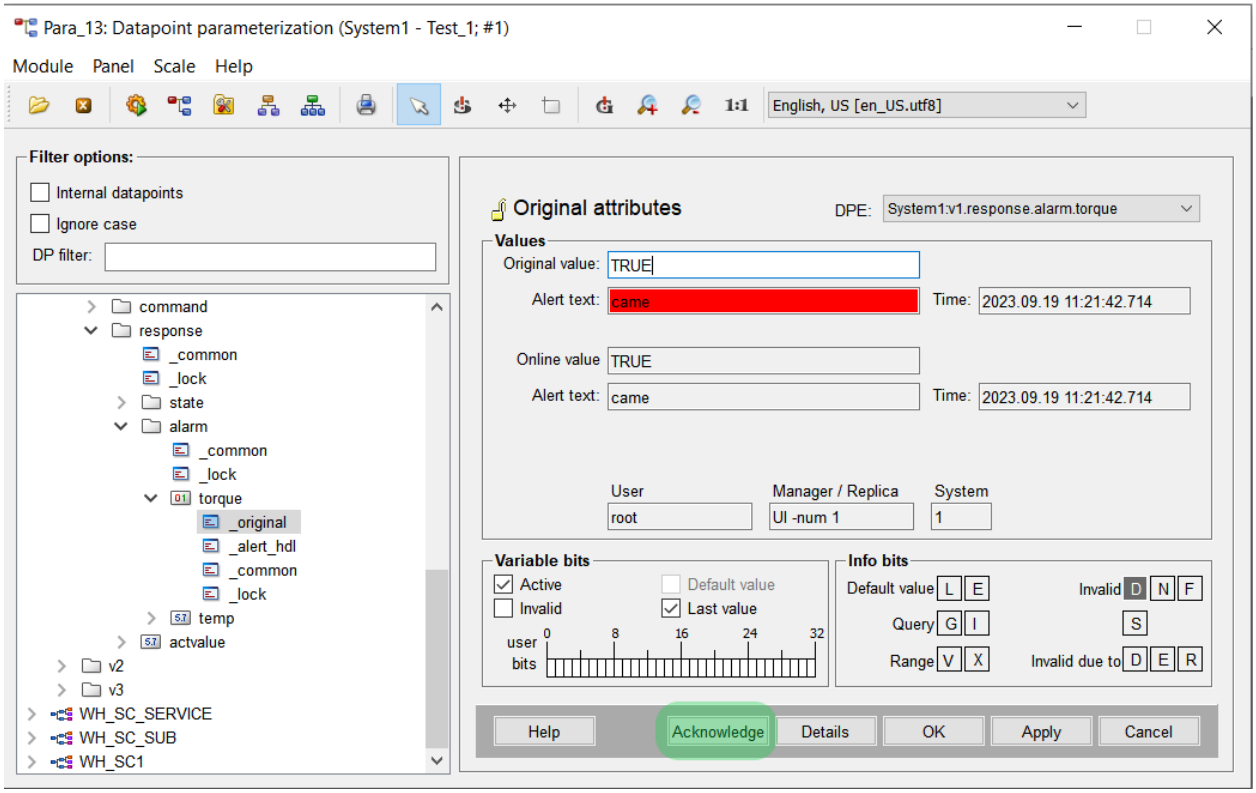
Класс аларма определяет многие аспекты отображения сигнализации: какой будет цвет, задействовано ли мерцание, будет ли звуковое уведомление, требуется ли

квитирование и т.д. Выбирите класс 230_S7_Alarm и поставьте галочку рядом со словом «Active». После выставления аларма активным все настройки становятся недоступны. Настройки допускается менять только для неактивных алармов.

Перейдем в модуле para на конфиг original, посмотрим текущее значение и поменяем его на «истину».

Вернем значение переменной в FALSE. Текст аларма остается «came», продолжает мигать, изменилась лишь интенсивность его мигания.

Данный класс алармов подразумевает необходимость квитирования. Для подтверждения аларма в этом окне необходимо нажать кнопку Acknowledge, после чего все вернется в «нормальное» состояние.



Сквитирование неушедшее аварийное сообщение перестаёт мигать, но продолажет подсвечиваться красным цветом до тех пор, пока состояние не восстановится (вернётся в норму).

2. ПРАКТИЧЕСКИЕ ЗАДАНИЯ И ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

Практическая работа №1. Создание и запуск проекта

Цель: научиться создавать новый проект APDAR, настраивать его параметры, запускать и останавливать менеджеры, анализировать состояние системы.

Задания:

1. Создайте новый стандартный проект с именем MyFirstProject на английском языке. Путь к проекту выберите самостоятельно (латинскими буквами, без пробелов). Откажитесь от пароля root.
2. Убедитесь, что проект появился в таблице администратора проектов. Запустите его через кнопку «светофор».
3. В консоли APDAR изучите список менеджеров. Определите, какие менеджеры запущены по умолчанию (EV, DM, UI). Запомните их номера.
4. Откройте config-файл проекта двойным кликом. Найдите в нём секции [general], [ui]. Изучите параметры pvss_path, langs. Убедитесь, что последняя строка файла не пуста (заканчивается нажатием Enter).
5. Остановите проект (закройте консоль или нажмите Stop). Перезапустите его снова.

Вопросы для самоконтроля:

1. Какие символы запрещено использовать в имени проекта APDAR?
2. Зачем нужен администратор проектов? Что такое PMON?
3. Назовите три основных менеджера APDAR и их функции (EV, DM, UI).
4. Что означает состояние менеджера «2»? «3»?
5. Почему важно, чтобы последняя строка config-файла была пустой?
6. Для чего используется параметр langs? Можно ли изменить порядок языков после создания проекта?

Практическая работа №2. Моделирование данных в редакторе PARA

Цель: освоить создание типов точек данных (DPT) и экземпляров (DP), настройку конфигов, понимание структуры DPE.

Задания:

1. Откройте редактор PARA. Создайте тип точки данных Motor (двигатель).
2. Добавьте в него следующие узлы (элементы):
    - command (группа)
        - start (тип bool)
        - stop (bool)

    - response (группа)
        - speed (float)
        - current (float)
        - running (bool)
3. Создайте три точки данных (экземпляра) типа Motor с именами m1, m2, m3.
4. Для DPE m1.response.speed откройте конфиг Archive settings и включите архивирование.
5. Для DPE m1.response.current добавьте конфиг Alert settings. Установите «хороший» диапазон от 0 до 50 А, текст аларма «Ток превышен», класс аларма 230_S7_Alarm. Активируйте аларм.
6. Для DPE m1.command.start настройте конфиг Default value со значением по умолчанию FALSE.
7. Изучите конфиг _original у любого DPE. Введите в него произвольное значение и нажмите Enter. Понаблюдайте за полем Online Value.

Вопросы для самоконтроля:

1. Чем отличается тип точки данных (DPT) от точки данных (DP)?
2. Что такое DPE (datapoint element)? Приведите пример.
3. Какие конфиги отвечают за архивирование и за аварийную сигнализацию?
4. Для чего нужен конфиг Periphery address? Почему он важен для лицензирования?
5. Что делает конфиг Smoothing? Когда его следует применять?
6. В чём разница между Original Value и Online Value?
7. Что означает элемент _lock в дереве DP?

Практическая работа №3. Создание мнемосхемы в GEDI

Цель: научиться создавать графические панели, использовать примитивы, настраивать свойства и анимацию.

Задания:

1. Откройте GEDI. В корне панелей (Panels) создайте новую панель с именем motor_panel.
2. Нарисуйте окружность (имитация двигателя) и два прямоугольника (имитация трубопровода или рамы).
3. Скопируйте окружность дважды, раскрасьте в разные цвета (например, зелёный – работа, красный – останов).
4. Добавьте две кнопки: «Пуск» и «Стоп». Измените их надписи через свойство Button label. Сделайте надписи многоязычными (русский/английский), скопировав на все языки.
5. Используя инструмент выравнивания, расположите элементы аккуратно.

6. Сохраните панель. Нажмите «Save and run», проверьте отображение в режиме исполнения.

Вопросы для самоконтроля:

1. Какие форматы панелей поддерживает APDAR? Какой предпочтительнее и почему?
2. Как изменить цвет фона графического объекта?
3. Как создать собственный цвет RGB в палитре APDAR?
4. Что такое опорная точка объекта (reference point)? Как её переместить?
5. Для чего используется инструмент выравнивания (Align)?
6. Как сделать так, чтобы кнопка выглядела одинаково на русском и английском языке?

Практическая работа №4. Динамика и подписка на изменения

Цель: освоить привязку графических объектов к точкам данных, использование функций dpSet, dpGet, dpConnect.

Задания:

1. В панели motor_panel выделите одну из окружностей. Через событие Initialize с помощью Wizard (Simple) настройте динамическое свойство Fill color в зависимости от DPE m1.response.running: при значении TRUE – зелёный цвет, при FALSE – красный.
2. Запрограммируйте кнопку «Пуск»: по событию Clicked установите DPE m1.command.start в TRUE с помощью dpSet.
3. Кнопку «Стоп»: установите m1.command.start в FALSE.
4. Создайте глобальный скрипт motor_sim.ctl со следующим поведением:
    - Подпишитесь на m1.command.start.
    - В callback-функции: если команда TRUE, то через цикл с задержкой (delay(0,50)) увеличивайте m1.response.speed от 0 до 100% и установите running = TRUE; если FALSE – уменьшайте скорость до 0, затем running = FALSE.
5. Добавьте в консоли новый менеджер Control с номером -num 2, укажите путь к скрипту, запустите вручную.
6. Проверьте: при нажатии «Пуск» окружность меняет цвет на зелёный, скорость растёт; при «Стоп» – краснеет, скорость падает.

Вопросы для самоконтроля:

1. Чем отличается dpSet от dpGet?
2. Что делает функция dpConnect? Какие аргументы она принимает?

3. Что такое callback-функция? Какие параметры ей передаются?
4. Зачем нужна функция delay() в имитации движения?
5. Почему после изменения глобального скрипта нужно перезапускать Control Manager?
6. Как в APDAR реализован событийный механизм (а не polling)?

Практическая работа №5. Шаблоны и доллар-параметры

Цель: научиться создавать параметризованные панели для многократного использования.

Задания:

1. Скопируйте панель motor_panel под именем motor_template.xml. Сохраните её в формате XML.
2. Откройте скрипты панели (Ctrl+E). Замените все вхождения m1 на dp (например, System1:m1.response.running → System1:dp.response.running). Знак + используйте для конкатенации строк: "System1:" + dp + ".response.running".
3. В комментариях, созданных Wizard (строка с //DP), также замените m1 на dp.
4. Создайте новую панель overview. Перетащите на неё три раза шаблон motor_template. При каждом добавлении в появившемся окне введите фактическое имя DP: m1, m2, m3.
5. Запустите панель overview в режим исполнения. Проверьте, что каждая задвижка (двигатель) управляется независимо.
6. Если скрипт motor_sim.ctl обслуживает только m1, модифицируйте его для работы с любым мотором, используя dpQueryConnectSingle.

Вопросы для самоконтроля:

1. Что такое доллар-параметр (dp)? Зачем он нужен?
2. Почему при создании шаблона рекомендуется сохранять панель в формате XML?
3. Как система узнаёт, что нужно заменить dp при вставке шаблона на панель?
4. Какая функция позволяет подписаться на все точки данных, удовлетворяющие запросу (query)?
5. Что делает функция dpSubStr с параметром DPSUB_SYS_DP?

Практическая работа №6. Рабочее место оператора и топология панелей

Цель: создать многопанельный интерфейс с навигацией, настроить стартовую панель.

Задания:

1. В GEDI откройте редактор топологии панелей (Panel Topology). Выберите 1 монитор, шаблон TEMPLATE1.
2. В дереве топологии создайте корневую панель main.pnl (если не создана автоматически).
3. Добавьте дочерний узел Overview с типом Root. Укажите панель overview.pnl (ту, что с тремя двигателями).
4. Добавьте дочерние узлы для каждого двигателя: Motor 1, Motor 2, Motor 3. Для каждого укажите панель-шаблон motor_template и в параметрах передайте dp = m1, m2, m3 соответственно. Тип панели – Child (всплывающее окно).
5. Добавьте в консоли новый менеджер User Interface с параметрами -p vision/startup.pnl -num 2. Запустите.
6. При входе под пользователем root (без пароля) должна открыться оболочка. Проверьте, что в навигационной панели появились кнопки для перехода к Overview и каждому двигателю. Настройте кнопки через контекстное меню Properties, привязав к соответствующим панелям в топологии.

Вопросы для самоконтроля:

1. Что такое «топология панелей»? Зачем она нужна?
2. Чем отличается Root-панель от Child-панели?
3. Как задать разрешение экрана для рабочего места оператора?
4. Какой файл является стартовой панелью (start panel) по умолчанию?
5. Каким образом можно добавить навигационную кнопку для открытия панели?

Практическая работа №7. Аварийные сообщения и их квитирование

Цель: настроить аварийную сигнализацию, научиться работать с конфигом Alert handling и отображать сообщения.

Задания:

1. В редакторе PARA для точки данных m1.response.current уже был настроен аларм. Откройте его конфиг Alert handling. Установите:
    - Good range: 0 – 50 (значения >50 – аларм)
    - Текст аларма: "Ток превышен" / "Current exceeded"
    - Класс аларма: 230_S7_Alarm

- Активен: да

2. В глобальном скрипте motor_sim.ctl добавьте имитацию роста тока при увеличении скорости: например, current = speed * 0.8. При speed > 60 ток будет >48, при speed > 63 – >50, что вызовет аларм.
3. Запустите проект. В оболочке оператора откройте панель алармов (должна присутствовать в шаблоне TEMPLATE1). Увеличьте скорость двигателя через кнопку «Пуск». Дождитесь срабатывания аларма.
4. Изучите поведение аларма: мигание, цвет, звук (если настроен). Выполните квитирование (Acknowledge) до возврата в норму. Обратите внимание, что квитированный непогашенный аларм продолжает гореть, но не мигает.
5. Снизьте скорость (кнопка «Стоп») ниже порога. Убедитесь, что аларм перешёл в состояние «норма». Сквитируйте его (если требуется).

Вопросы для самоконтроля:

1. От каких типов переменных зависит внешний вид окна Alert handling?
2. Что означает «хороший диапазон» (Good range) для аналогового аларма? Что будет при выходе значения из этого диапазона?
3. Что такое класс аларма (Alert class)? Какие атрибуты он определяет?
4. В чём разница между «активным» и «неактивным» алармом?
5. Что происходит при квитировании аларма, если он всё ещё активен? А если он уже вернулся в норму?
6. Где в проекте можно просмотреть журнал аварийных сообщений?

Практическая работа №8. Мониторинг технического состояния коллаборативного робота-манипулятора

Цель: Реализовать человеко-машинный интерфейс системы диагностирования технического состояния коллаборативного робота, входящего в состав гибкой производственной системы (ГПС) на основе контрольно-измерительной информации, поступающей из управляющей системы робота.

Основой регистрации технического состояния оборудования ГПС является определение фактических значений его основных характеристик (параметров) и сопоставление их с номинальными значениями. В результате сопоставления характеристик (параметров) выявляют симптомы (внешние признаки) текущего состояния компонентов ГПС. При диагностировании оценивают результаты сопоставления этих состояний либо непрерывно в течение всего времени эксплуатации оборудования, либо периодически (через малые промежутки времени, нормированные длительные промежутки времени в ходе профилактического обслуживания или случайные промежутки времени при отказах) [9, 10]. В таблице 2.1 представлены параметры коллаборативного робота (кобота) Elite Robots CS63 характеризующие его состояние в процессе функционирования. Получение/установка данных параметров управляющей системы кобота осуществляется

по протоколу Modbus TCP. Модель CS63 от ELITE ROBOTS — это компактный 6-осевой кобот с грузоподъёмностью до 3 кг и высокой точностью позиционирования ±0,02 мм предназначен для решения таких задач, как сборка небольших изделий, транспортировка материала на более высокий уровень, загрузка/выгрузка деталей в станки для их обработки, контроль качества, сверление, завинчивание, нанесение герметика, сортировка.

Таблица 2.1 – Параметры коллаборативного робота ELITE ROBOTS CS63

Номер параметра	Адрес Modbus регистра	Чтение /Запись	Описание
1	0	R	Стандартные входы (BBBBBBBBBBBBBBB)
2	1	R	Конфигурируемые входы и инструмент (xxxxTTTTCCCCCCCC)
3	2	R/W	Стандартные выходы (BBBBBBBBBBBBBBB)
4	3	R/W	Конфигурируемые выходы и инструмент (xxxxTTTTCCCCCCCC)
5	4	R	Аналоговый вход [0] (мА; мВ)
6	5	R/W	Поле аналогового входа [0] (0:Ток; 1:Напряжение)
7	6	R	Аналоговый вход [1] (мА; мВ)
8	7	R/W	Поле аналогового входа [1] (0:Ток; 1:Напряжение)
9	8	R	Аналоговый вход [2](инструмент) (мА; мВ)
10	9	R/W	Поле аналогового входа [2](инструмент) (0:Ток; 1:Напряжение)
11	10	R/W	Аналоговый выход [0] (мА; мВ)
12	11	R/W	Поле аналогового выхода [0] (0:Ток; 1:Напряжение)
13	12	R/W	Аналоговый выход [1] (мА; мВ)
14	13	R/W	Поле аналогового выхода [1] (0:Ток; 1:Напряжение)
15	14	R/W	Аналоговый выход (инструмент) [2] (мА; мВ)
16	15	R/W	Поле аналогового выхода (инструмент) [2] (0:Ток; 1:Напряжение)
17	16	R/W	Выходное напряжение инструмента (В) (0 В 12 В)
18	63	R	Основная версия контроллера
19	64	R	Вспомогательная версия контроллера
20	65	R	Версия отладки контроллера
21	66	R	Режим робота (см. мануал по Modbus client)
22	67	R	Есть ли питание на роботе
23	68	R	Включён ли защитный останов
24	69	R	Включён ли аварийный останов
25	70	R	Включён ли замедленный режим
26	71	R	Режим управления
27	73	R	База угла поворота сочленений (mRad)
28	74	R	Угол поворота плечевого звена (mRad)
29	75	R	Угол поворота локтевого звена (mRad)
30	76	R	Угол поворота запястья 1 (mRad)
31	77	R	Угол поворота запястья 2 (mRad)
32	78	R	Угол поворота запястья 3 (mRad)

Номер параметра	Адрес Modbus регистра	Чтение /Запись	Описание
33	84	R	База скорости сочленений (mRad/s)
34	85	R	Скорость плечевого звена (mRad/s)
35	86	R	Скорость локтевого звена (mRad/s)
36	87	R	Скорость запястья 1(mRad/s)
37	88	R	Скорость запястья 2(mRad/s)
38	89	R	Скорость запястья 3(mRad/s)
39	95	R	Общий ток базы (мА)
40	96	R	Общий ток плечевого звена (мА)
41	97	R	Общий ток локтевого звена (мА)
42	98	R	Общий ток запястья 1 (мА)
43	99	R	Общий ток запястья 2 (мА)
44	100	R	Общий ток запястья 3 (мА)
45	105	R	Общая температура базы (°C)
46	106	R	Общая температура плечевого звена (°C)
47	107	R	Общая температура локтевого звена (°C)
48	108	R	Общая температура запястья 1 (°C)
49	109	R	Общая температура запястья 2 (°C)
50	110	R	Общая температура запястья 3 (°C)
51-	256-383	R/W	Регистры общего назначения

R – чтение; W – запись.

Задание:

1. Создайте новый проект.
2. Для каждого параметра кобота, представленного в таблице 2.1 в редакторе PARA создайте точку данных и при необходимости настройте аларм используя конфиг Alert handling. Значения параметров кобота являются внешними тегами проекта и устанавливаются через конфиг Periphery address.
3. Для всех созданных точек данных настройте конфиг Archive settings обеспечив запись их значений в историческую базу данных проекта.
4. Используя функциональные возможности редактора GEDI разработайте мнемосхему отображающую текущее положение звеньев кобота, их температуры, состояния входов/выходов, режимов работы кобота.
5. Отладьте созданный проект используя тестовую программу для кобота. Выбор программы и соответствующей конфигурации кобота выполняется с помощью пульта управления с сенсорным экраном, входящего в его состав.
6. По результатам выполнения задания подготовьте отчёт.

Итоговые контрольные вопросы (для самопроверки)

1. Опишите полный жизненный цикл значения точки данных от внешнего устройства до отображения на мнемосхеме.
2. Какие способы передачи данных между скриптами и графикой существуют в APDAR?
3. Почему APDAR называют событийно-ориентированной SCADA-системой? Приведите пример.
4. Каковы основные отличия APDAR от других SCADA-систем (например, WinCC, Citect, MasterSCADA)?
5. Что произойдёт, если в config-файле последняя строка не будет пустой?
6. Как можно реализовать звуковое оповещение при аварии?
7. Как ограничить доступ оператора к определённым элементам управления (например, к задвижке v3)?
8. Что такое «менеджер сценариев» (Control Manager) и почему он запускается отдельно?
9. Какие инструменты APDAR предназначены для отладки скриптов?
10. Как создать новый язык интерфейса, если изначально был выбран только русский?

Заключение

В ходе выполнения практических работ, описанных в методических указаниях, обучающийся последовательно прошёл все основные этапы создания проекта автоматизации в SCADA-системе APDAR: от инициализации проекта и настройки менеджеров до построения полнофункционального рабочего места оператора с динамической графикой, обработкой команд и аварийной сигнализацией.

В результате освоения материала были сформированы следующие практические навыки и компетенции:

1. Работа с инфраструктурой проекта – создание, запуск, конфигурирование проекта; понимание роли менеджеров (EV, DM, Control, UI, Archive) и их взаимодействия; умение читать и редактировать config-файл.

2. Моделирование данных – создание типов точек данных (DPT) и экземпляров точек данных (DP) в редакторе Para; настройка конфигурационных узлов (_original, _online, alert handling, periphery address и др.); понимание различий между Original Value и Online Value.

3. Разработка графического интерфейса – создание мнемосхем в GEDI с использованием графических примитивов; настройка динамических свойств (поворот, цвет, видимость); управление объектами через скрипты.

4. Программирование логики управления – написание скриптов на встроенном языке APDAR с использованием функций dpGet, dpSet, dpConnect; реализация механизма подписки на изменения (callback-функции); создание глобальных скриптов для симуляции поведения оборудования (например, имитация движения задвижки с временной задержкой).

5. Повторное использование и масштабирование – создание параметризованных шаблонов панелей с доллар-параметрами ($dp); вызов шаблонов с подстановкой конкретных точек данных; использование dpQueryConnectSingle для динамической подписки на множество однотипных точек данных.

6. Организация рабочего места оператора – построение топологии панелей (panel topology); настройка корневой панели main.pnl, навигационных кнопок; вход в систему с аутентификацией пользователя (root).

7. Обработка аварийных сообщений – конфигурирование alert handling для дискретных сигналов (например, torque); настройка класса аларма, текста сообщения, квитирования; наблюдение за поведением алармов в реальном времени.

Полученные знания являются фундаментом для дальнейшего изучения более сложных аспектов APDAR: подключения реальных контроллеров через драйверы (OPC, Modbus RTU, Modbus TCP, S7), использования архивов и трендов, настройки пользовательских прав и многоязычности, оптимизации производительности и отказоустойчивости.

Рекомендуемая литература

1. Юсупов Р.Х. Основы автоматизированных систем управления технологическими процессами: учебное пособие /Р.Х. Юсупов. - 2-е изд. - Москва; Вологда: Инфра-Инженерия, 2025. - 132 с.
2. Тугов В.В., Сергеев А.И., Шаров Н.С. Проектирование автоматизированных систем управления: учебное пособие/В.В. Тугов, А.И. Сергеев, Н.С. Шарапов. - 3-е изд., стер. - Санкт-Петербург: Лань, 2022. - 172 с.
3. Шишов О.В. Технические средства автоматизации и управления: учебное пособие/О.В. Шишов. - М.: ИНФРА-М, 2026. - 396 с.
4. Бухарбаев М.А., Казагачев В.Н. Реализация алгоритмов управления и визуализации данных в SCADA-системах: учебное пособие/М.А. Бухарбаев, Н.В. Казагачев. - Алматы: ADAL KITAP, 2025. -176 с.
5. Иванов В.Э. Практическое проектирование систем управления в SCADA TRACE MODE 6: учебное пособие/В.Э. Иванов. - М.; Вологда: Инфра-Инженерия, 2025. - 232 с.
6. Иванов В.Э., Чье Ен Ун. Разработка АСУТП в среде WinCC: учебное пособие/В.Э. Иванов, Чье Ен Ун. - 2- е изд. - М.; Вологда: Инфра-Инженерия, 2025. - 232 с.
7. Кангин В.В., Кангин М.В., Ямолдинов Д.Н. Разработка SCADA-систем: учебное пособие/В.В. Кангин, М.В. Кангин, Д.Н. Ямолдинов. - 2-е изд. - М.; Вологда: Инфра-Инженерия, 2024. - 564 с.
8. Шишов О.В. Современные средства АСУ ТП: учебник/О.В. Шишов. - 2-е изд. - М.; Вологда: Инфра-Инженерия, 2025. - 532 с.
9. Автоматизация производственных процессов в машиностроении: Учеб. для втузов/ Н.М. Капустин, П.М. Кузнецов, А.Г. Схиртладзе и др.; Под ред. Н.М. Капустина. – 2-е изд., стер. – М.: Высш. Шк., 2007. – 415 с.
10. Брюханов Н.В. Автоматизация производства: учеб. для сред. проф. учеб. заведений/ Н.В. Брюханов, А.Г. Схиртладзе, В.П. Вороненко; под ред. Ю.М. Соломенцева. – М.: Высш. шк., 2005. – 367 с.