Руководство по эксплуатации среды разработки тестовых последовательностей «Itelma EOL Designer»

Редакция 2.1

Дата: 20.09.2025

1 Оглавление

2 Введение....4
3 Глоссарий....4
4 Общее представление системы EOL....4
5 Главное окно EOL Designer....5
5.1 Запуск EOL Designer....5
6 Строка меню EOL Designer....7
6.1 Меню File....7
6.2 Меню Build....8
6.3 Меню Settings....8
6.4 Меню Log In....9
6.5 Меню SVN....9
6.6 Меню Help....10
7 Секция проектов тестовых программ и файлов....10
7.1 Вкладка *Project*....10
7.2 Вкладка *Resources*....11
8 Секция выбора примитивов....12
8.1 Секция формирования тестовой последовательности....12
8.1.1 Base Parameters....13
8.1.2 Log Parameters....14
8.1.3 Specific Parameters....14
9 Логика работы и взаимодействия между примитивами в тестовой последовательности....15
10 Примитивы....17
10.1 Basic....17
10.1.1 Sequence Block....17
10.1.2 Delay....20
10.1.3 Print Log....21
10.1.4 Write Log....22
10.1.1 Random....23
10.1.2 Timer Start....24
10.1.3 Timer Check....24
10.1.4 XML Configurator....25
10.1.5 Set Vehicle....26
10.1.6 Project Block....29
10.1.7 Text Block....30

10.1.8 Get Section Name....30
10.1.9 Printer Log Start....31
10.1.10 Get Printer Log....33
10.1.1 Get Seed....36
10.2 Bench....36
10.2.1 Bench Report....36
10.2.2 Bench Show Message....37
10.2.3 Wait Synchro Word....37
10.2.4 Write Synchro Word....38
10.3 Current Clamp....39
10.3.1 Current Clamp Connect....39
10.3.2 Current Clamp Start....40
10.3.3 Current Clamp Stop....42
10.4 DEV File....44
10.4.1 Mnemonic Read....44
10.4.2 Mnemonic Write....45
10.4.3 Get VIN....47
10.5 Expressions....47
10.5.1 If Statement....48
10.5.2 Expression....53
10.6 Protocols....55
10.6.1 ODX Message....55
10.6.2 UDS Request....58
10.6.3 Seed and Key....60
10.6.4 DTC Read....61
10.6.5 StartComm....64
10.6.6 StopComm....64
10.7 Terminal....65
10.7.1 Operator Show Read....65

2 Введение

EOL Designer - это инструмент (среда программирования), который предназначен для создания необходимых тестовых программ (последовательностей);

В данном документе представлена основная информация для использования EOL Designer: его место в системе EOL, взаимодействие EOL Designer с компонентами системы EOL, необходимые условия для формирования тестовых последовательностей.

Описание и примеры сформированы для EOL Designer v1.25.1.

3 Глоссарий

ЭБУ	Электронный Блок Управления
EOL	End of Line / программно-аппаратный комплекс управления данными и процессами настройки ЭБУ и диагностики электрики и электрооборудования.
TSE	Test Sequence Executor / Программа, установленная на компьютере чек-станции, воспроизводящая тестовую последовательность.
DLL	Dynamic Link Library/Библиотека динамической компоновки данных.
Примитив	Один шаг тестовой последовательности EOL.
UDS	Unified diagnostic services/диагностической протокол передачи данных
Build	Процедура сборки и генерирования проекта в готовую тестовую последовательность в формате xml.
Main sequence	Главная тестовая последовательность проекта.
Active project	Акктивый проект.
SVN server/ сервер	Система управления версиями проекта

4 Общее представление системы EOL

Общее схема взаимодействия компонентов системы EOL представлено на рисунке 4.1

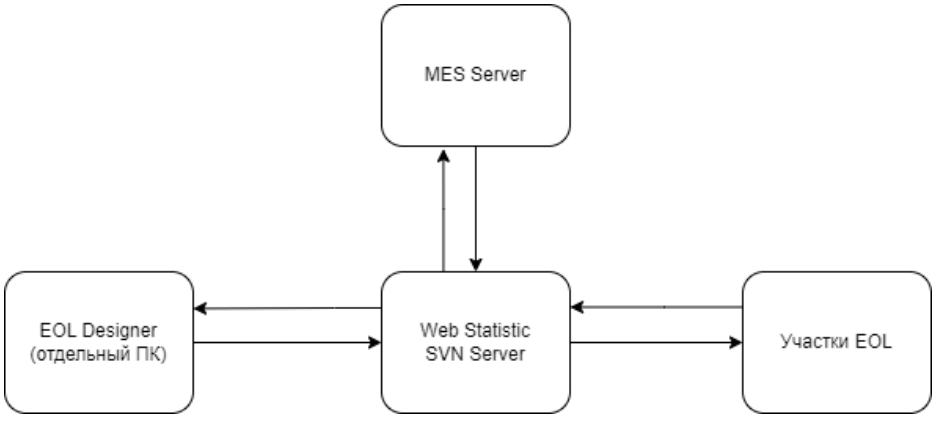


Рисунок 4.1 – Компоненты системы EOL.

Для работы в EOL Designer необходимо завести учетную запись с помощью Web Statistic.

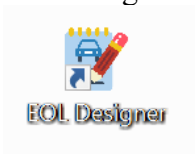
Просмотр учётных записей, редактирование и удаление также осуществляется через Web Statistic.

Описание Web Statistic представлено в документе «Руководство по использованию Web Statistic.docx».

В качестве системы контроля версий используется SVN Server.

5 Главное окно EOL Designer

5.1 Запуск EOL Designer

Запуск EOL Designer осуществляется двойным кликом по ярлыку приложения на рабочем столе 

.

После запуска необходимо пройти процедуру аутентификации пользователя: вести логин и пароль (рисунок 5.1).

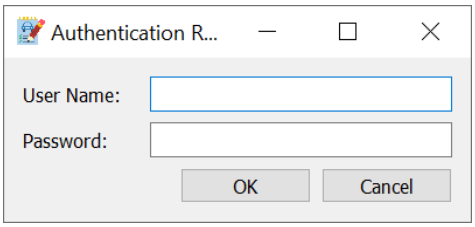


Рисунок 5.1 – Процедура аутентификации пользователя.

Если процедура аутентификации не прошла, то функционал EOL Designer будет ограничен. В ограниченном режиме пользователь имеет возможность редактировать настройки подключения к серверу (вкладка *Settings*), а также еще раз пройти процедуру аутентификации (вкладка *Log In*).

Если процедура аутентификации пользователя прошла успешно, то доступен полный функционал EOL Designer (рисунок 5.2).

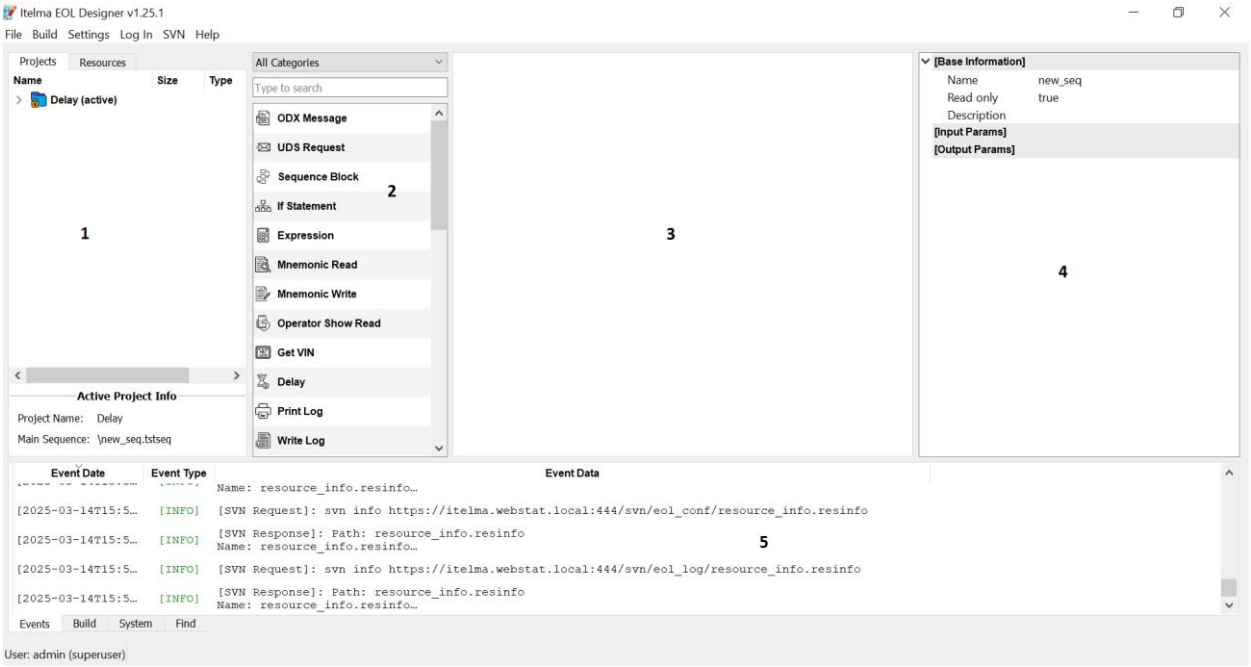


Рисунок 5.2 – Главное окно EOL Designer.

1 – секция проектов тестовых программ (Projects), секция подгружаемых файлов DTC, конфигураций, калибровок (Resources);
В данной секции есть поле Project Info в котором отображается имя активного проекта (Project Name) и главная тестовая последовательность в нем (Main Sequence).

2 – секция выбора примитивов;
В данной секции есть возможность фильтрации примитивов по категориям и по имени.

3 – секция формирования тестовой последовательности;

4 – краткое отображение характеристик выбранного примитива;

5 – информационное окно;
В данной секции имеется четыре вкладки*: Events, Build, System, Find*.

- Вкладка *Event* отображает лог последних действий разработчика тестовых последовательностей. Для данной вкладки характерны три типа лога (*Event Type*): *Info*, *Warning*, *Error*.

Если нет соединения с сервером, то появляется ошибки, где является адресом сервера: `[ERROR] Server response error: Host webstat.local not found`
При попытке запуска процедуры *Built* забокированного проекта выявляется ошибка: `[ERROR] Please unlock the project before building`
Если есть ошибки в тестовой последовательности, то при попытке запуска процедуры *Built* выявляются ошибки:

```
[ERROR]  BUILD STEP: Test sequences...ERROR

[ERROR]  Project build error
```

- Вкладка *Build* отображает лог сборки проекта. Для данной вкладки характерны четыре типа лога (*Event Type*): *Info*, *Warning*, *Error*, *Result*.

Если есть ошибки в формировании примитива, то после сборки в логе отображаются ошибки:

```
[ERROR]  Step param check error. [Test Sequence]: \new_seq.tstseq [Step]: 4. ODX Message [Param]: ODX_FileName [Error]: ODX file is not set
[ERROR]  [Step]: 4.ODX Message build ERROR
```

Если тестовая последовательность не прошла процедуру *Build,* то выявляется ошибка:

```
[ERROR]  [Test sequence]: new_seq.xml build error
```

Если в тестовой последовательности есть ошибки, связанные с линкованием, то отображается лог (лог показан на примере конкретного примитива):

```
[ERROR]  Linked param reference error. [Test sequence]: \new_seq.tstseq [Step]: 4.If True == "x1==x2" [Param]: x1
```

При двойном нажатии левой кнопкой мыши по тексту ошибки система подсветит некорректный примитив в секции формирования тестовой последовательности.

Если тестовая последовательность составлена без критических ошибок, но нарушен общий регламент последовательности, то система отобразит предупреждение:

```
[WARNING] Vehicle's name is not set into test sequence new_seq
```

- Вкладка *System* отображает системный лог необходимый для разработчиков среды EOL Designer.
- Вкладка *Find* позволяет осуществить поиск необходимых файлов в выбранном проекте или во всех проектах. А также позволяет найти текст с в тестовых последовательностях выбранного проекта.

В нижнем левом углу главного окна приложения отображается имя текущего пользователя и прогресс бар. Прогресс бар визуально отображает ход выполнения задачи.

6 Строка меню EOL Designer

Строка меню располагается в верхней части главного окна приложения. Пункты меню: File, Build, Settings, Log In, SVN, Help.

6.1 Меню File

Данный раздел содержит операции с проектами:

- *New Project* позволяет создать новый проект;

- *New Offline Project* позволяет создать проект локально на ПК разработчика. Для того, чтобы выгрузить проект на *SVN server* необходимо кликнуть правой кнопкой мыши по корневой папке проекта и из выпадающего меню выбрать *Export to SVN server*.
- *Clone Project* позволяет скопировать последний релиз проекта локально на компьютер;
- *Fork Project* позволяет создать новую ветку проекта, на базе любого релиза выбранного проекта;
- *Open Project* позволяет открыть проект с внутренних источников;
- *Save Project* позволяет сохранить проект локально на компьютере.
- *Import Project* позволяет импортировать проект из внешних источников;

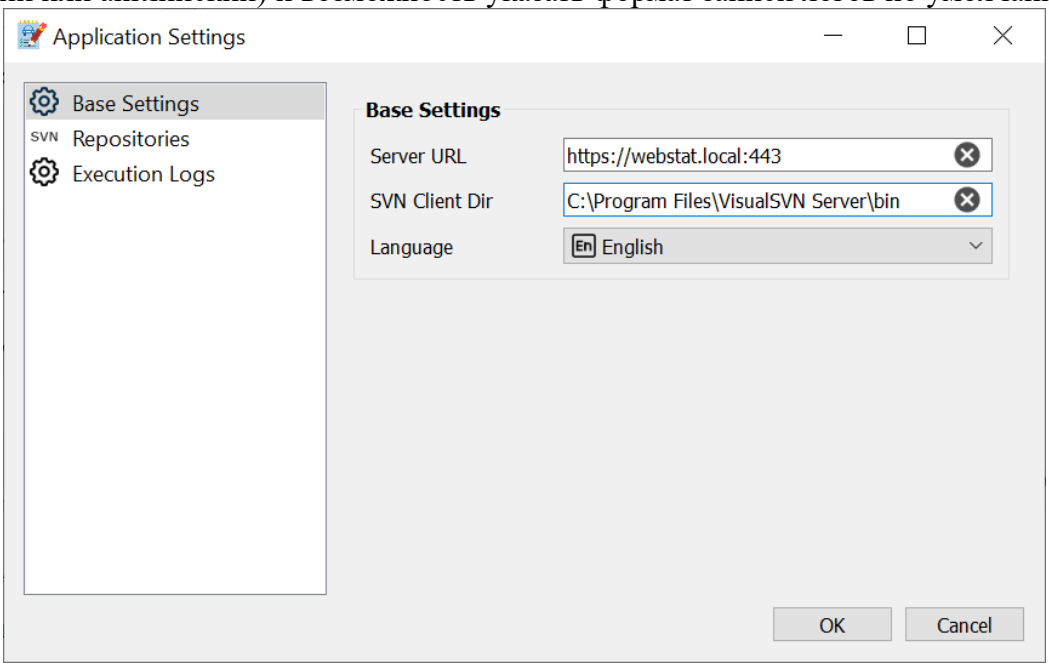
6.2 Меню Build

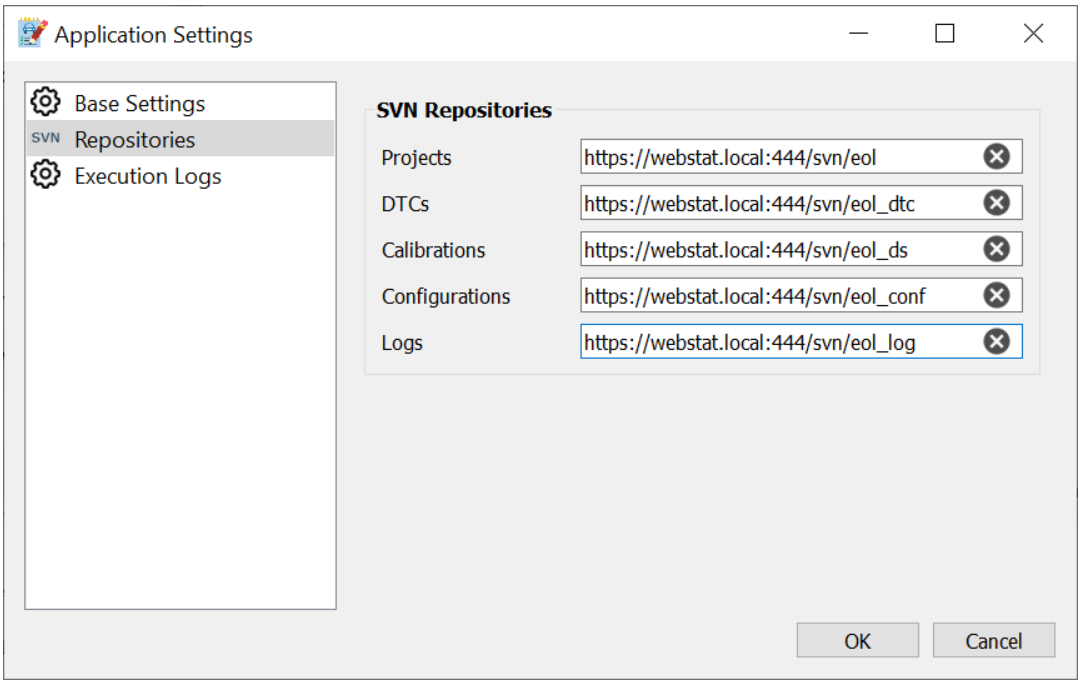
Build позволяет сгенерировать необходимый проект и сохранить текущий вариант тестовой последовательности в систему контроля версий.

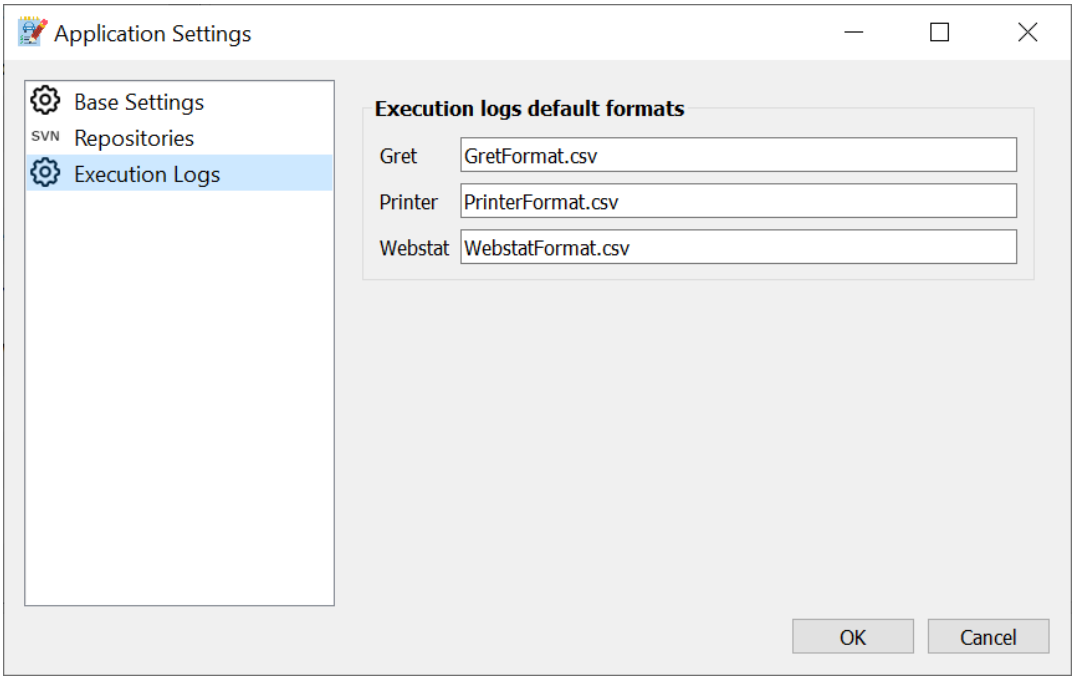
Для генерации проекта необходимо выбрать активный проект путем нажатия правой кнопкой мыши по проекту и выбора из выпадающего меню *Active project.* Также в проекте должна быть выбрана *Main Sequence*.

6.3 Меню Settings

В данном разделе осуществляется настройка путей хранения проектов на сервере, DTC, калибровок, конфигураций и логов. Также предлагается выбор языка приложения (русский или английский) и возможность указать формат записи логов по умолчанию.







6.4 Меню Log In

В данном разделе пользователь имеет возможность поменять учетную запись пользователя.

6.5 Меню SVN

Данный раздел содержит следующие операции:

- Update Projects – синхронизация всех проектов с их последней ревизией на сервере.
- Update Resources – синхронизация всех проектов с их последней ревизией на сервере.

6.6 Меню Help

Help предоставляет основную информацию о поставщике приложения.

7 Секция проектов тестовых программ и файлов

Данная секция обозначена цифрой **1** на рисунке 5.2 и содержит две вкладки: *Project* и *Resources*.

7.1 Вкладка *Project*

Project отображает текущие проекты. Операции над проектами доступные при нажатии правой кнопкой мыши по проекту:

- *Active Project* – выбор активного проекта, который будет проходить процедуру *build* при ее запуске. Также в активном проекте необходимо указать *main sequence*.
- *Close Project* – локальное закрытие проекта.
- *Unlock to edit* – разблокировка проекта для внесения изменений. В данном случае при разблокировке проекта для редактирования на сервере осуществляется его блокировка, т.е. любой другой пользователь (кроме администратора и разработчика с расширенными привилегиями) не сможет вносить изменения в данный проект.
- *Lock and commit* – позволяет сохранить текущие изменения проекта в системе контроля версий. После данной процедуры проект выгружается на сервер с назначенным системой номером ревизии и блокируется для редактирования.
- *Update* – синхронизация файлов проекта с последней ревизией на сервере (если локально были созданы дополнительные файлы, то при процедуре *update* они не удаляются).
- *History* – просмотр истории изменений проекта.
- *Status* – отображает состояние файлов и каталогов рабочей копии проекта. Под рабочей копией проекта подразумевается локально сохраненный проект. Данная операция необходимо разработчикам приложения для анализа возникающих проблем. Более подробная информация по ссылке https://svnbook.red-bean.com/en/1.8/svn.ref.svn.c.status.html
- *Reset Changes* – отмена последних изменений с момента последнего сохранения.
- *Cleanup* – очистка.
- *Forced unlock* – это процедура разблокировки проекта в то время, когда он заблокирован другим пользователем. Воспользоваться данной процедурой могут только разработчики с повышенными привилегиями и администратор.

Для корректной работы процедуры *forced unlock* необходимо осуществить следующую последовательность процедур: *Forced unlock -> Reset changed -> Update.*

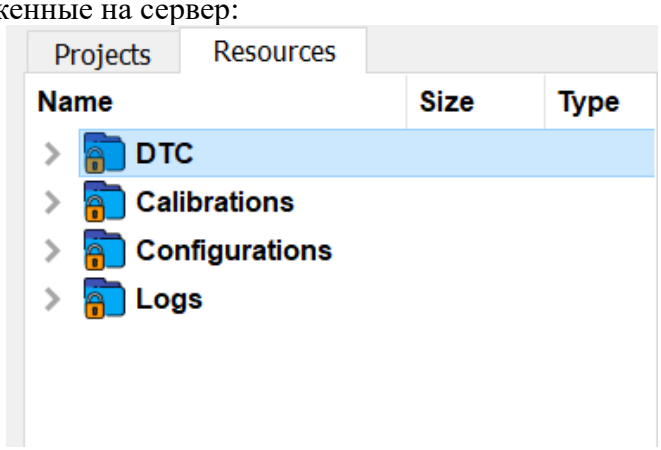
Каждый проект может содержать несколько тестовых последовательностей, но одна из них должна быть отмечена как главная (*main sequence*). Готовый исполнительный файл тестовой последовательности будет генерироваться из *main sequence.*

Операции над файлом тестовой последовательности доступные при нажатии правой кнопкой мыши по проекту:

- *Edit Parameters* позволяет просматривать тип последовательности: *main sequence*, *read only*, добавлять описание к файлу в поле *Description*, а также добавлять входные и выходные параметры последовательности.
- *Refresh* – обновить.
- *Sat as Main Sequence* позволяет назначить выбранную последовательность как главную (*mail sequence*). Соответственно в сгенерированном *xml* файле будут содержаться примитивы из этой последовательности и вложенные в данную последовательность подпрограммы.
- *Create Sequence* позволяет создать новую тестовую последовательность.
- *Create Folder* позволяет создать новую папку.
- *Copy* копирует выбранную тестовую последовательность;
- *Past* вставляет скопированный файл тестовой последовательности в выбранную папку.
- *Rename* позволяет изменить имя выбранной тестовой последовательности.
- *Delete* удаляет выбранную тестовую последовательность.

7.2 Вкладка *Resources*

В данной вкладке отображены файлы DTC, калибровки и конфигурации, и логи загруженные на сервер:



Для разделов DTC, Calibrations, Configurations и Logs пользователь имеет возможность загружать дополнительные файлы на сервер и удалять их.

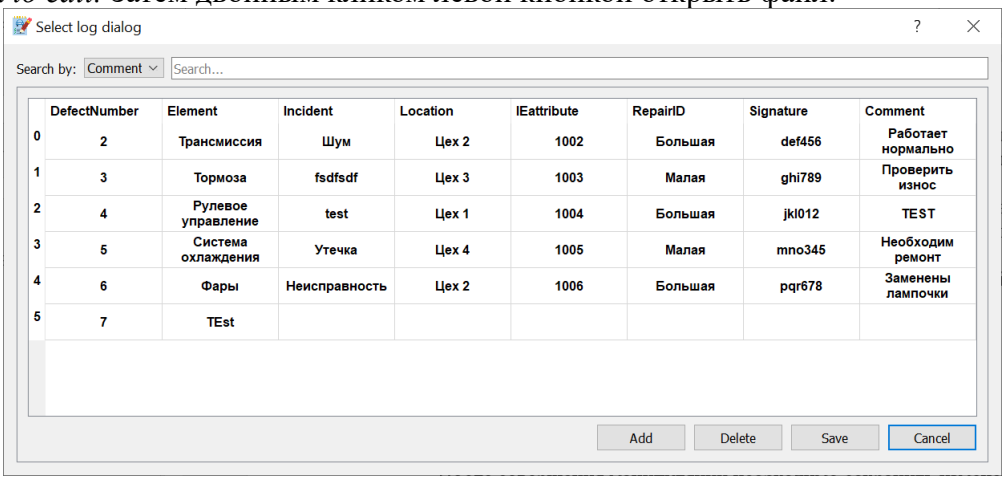
Для реализации манипуляций необходимо разблокировать нужный раздел путем нажатия правой кнопки мыши по разделу и выбора из выпадающего меню *Unlock to edit.*

Операции над файлами раздела доступные при нажатии правой кнопкой мыши по любому файлу внутри выбранного раздела:

- *Refresh* – обновить файл;
- *Create Folder* – создать новый раздел;
- *Add Files* – добавить файл;
- *Delete* – удалить файл;
- *Rename* – переименовать файл.

После совершения манипуляций необходимо сохранить изменения путем нажатия правой кнопкой мыши по разделу с изменениями и выбра из выпадающего меню *Lock and commit*.

Для раздела Logs кроме опреций над файлами доступна возможность редактирования файла. Для этого необходимо разблокировать раздел через процедуру *Unlock to edit.* Затем двойным кликом левой кнопкой открыть файл.



Внести необходимые изменения и сохранить, нажав кнопку *Save*. Для фиксирования изменений на сервере необходимо совершить процедуру *Lock and commit.*

8 Секция выбора примитивов

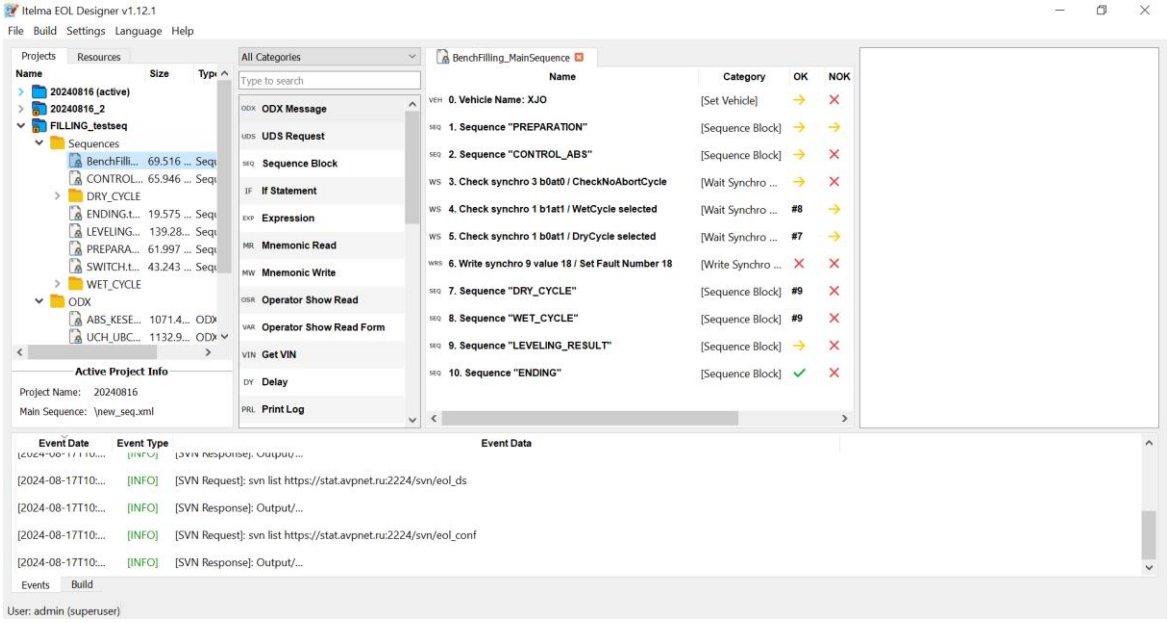
Данная секция обозначена цифрой **2** на рисунке 5.2**.** Примитивы разделяются на несколько категорий:

- Basic;
- Bench;
- Current Clamp;
- DEV File;
- Expressions;
- Protocols;
- Terminal.

8.1 Секция формирования тестовой последовательности

Данная секция обозначена цифрой **3** на рисунке 5.2**.** В данной секции разработчик формирует тестовую последовательность.

Для открытия окна необходимой тестовой последовательности нужно в секции тестовых программ (секция 1, рисунок 5.2) выбрать нужный файл и нажать по нему двойным кликом. Таким образом его содержимое отобразится в секции 3 (рисунок 5.3). Из секции примитивов пользователь перетаскивает объекты и располагает в нужной последовательности.



Окно с настройками выбранного примитива открывается двойным нажатием левой кнопкой мыши по примитиву в секции формирования тестовой последовательности, либо однократным нажатием правой кнопкой мыши и затем из выпадающего меню выбором *Edit Properties*.

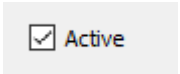
Окна *Base Parameters* и *Log Parameters* являются общими для всех примитивов.

Примечание: для примитива *ODX Message* окно *Base Parameters* имеет небольшое отличие.

Также при нажатии правой кнопкой мыши по примитиву доступны стандартные операции такие как: *Copy*, *Paste*, *Undo* (отмена последнего действия), *Redo* (возврат действия, отмененного вызовом *Undo*), *Delete*.

8.1.1 Base Parameters

Окно Base Parameters содержит основные характеристики примитива. Галочка



Active позволяет включить примитив в список воспроизведения или закомментировать (выключить).

Step Information

Название поля	Описание
Step Custom Name	В данном поле разработчик может задать имя примитива, которое будет отображаться в окне тестовой последовательности.
Step Name	Содержит название примитива. Заполняется системой автоматически.
Step ID	Идентификатор примитива. Заполняется системой автоматически.

Step Category	Содержит название категории, к которой принадлежит примитива. Заполняется системой автоматически.
Step Subcategory	Заполняется системой автоматически.
Step Description	Данное поле предназначено для комментариев разработчика.

Step Targets описан в разделе Логика работы и взаимодействия между примитивами в тестовой последовательности.

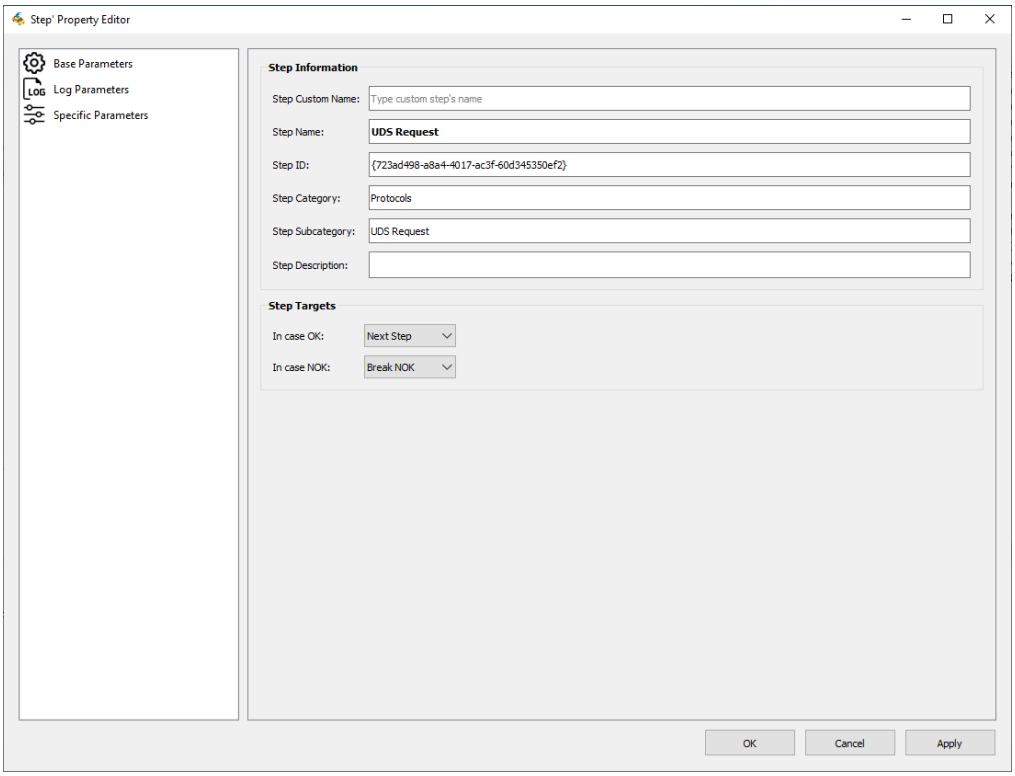


Рисунок 8.1 – Пример окна Base Parameters.

8.1.2 Log Parameters

В данном окне настраиваются параметры записи лога. Лог может быть записан в систему *Gret*, распечатан на принтере (*Printer*) и зафиксирован в *Web Statistic*. Формат записи лога выбирается из выпадающего меню.

Разработчик имеет возможность выбора записи лога в зависимости от результата примитива (о результатах примитива подробнее написано в пункте Логика работы и взаимодействия между примитивами в тестовой последовательности).

8.1.3 Specific Parameters

Окно Specific Parameters содержит специфические параметры для каждого примитива.

9 Логика работы и взаимодействия между примитивами в тестовой последовательности

Примитив представляет из себя один шаг тестовой последовательности, который имеет входные и выходные параметры (рисунок 9.1). Функционал примитивов более подробно описан в разделе 10.



Рисунок 9.1 – Общее преставление примитива.

Input Parameters и *Output Parameters* являются опциональными и зависят от функционала примитива.

Step Result является обязательным выходным параметром и формируется в соответствии с таблицей 9.1:

Step Target		Step Result
Результат функции (In case)	Выбор шага	
OK	Next Step	true
NOK	Next Step	true
OK	Break NOK	false
NOK	Break NOK	false
OK	Break OK	true
NOK	Break OK	true
OK	Custom Step	true
NOK	Custom Step	true

Таблица 9.1 – Таблица истинности для Step Result.

Под результатом функции подразумевается внутренний результат функции примитива. Например, внутренний результат функции примитива *UDS request* зависит от ответа ЭБУ. Если ответ отрицательный (*negative response),* то внутренний результат функции примитива *NOK*, если положительный *(positive response*), то *OK*. При этом, в зависимости от выбора следующего шага тестовой последовательности зависит *Step Result* самого примитива.

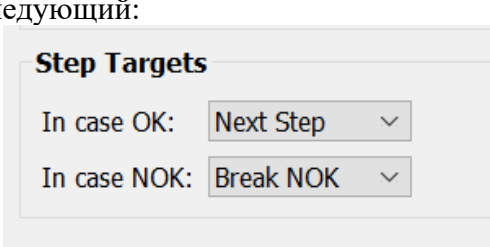
Выполнение тестовой последовательности начинается с нулевого примитива. Далее, в зависимости от настроек примитива, определяется следующий шаг, к которому перейдет программа. Настройки следующего шага осуществляются в окне *Base Parameters* в поле *Step Targets* каждого примитива.

Step Targets определяет следующий шаг программы в зависимости от внутреннего результата функции примитива. Результат может быть *OK* и *NOK*. Для каждого значения *Step Target* необходимо выбрать к какому следующему шагу будет переходить тестовая последовательность.

Доступны следующие варианты перехода:

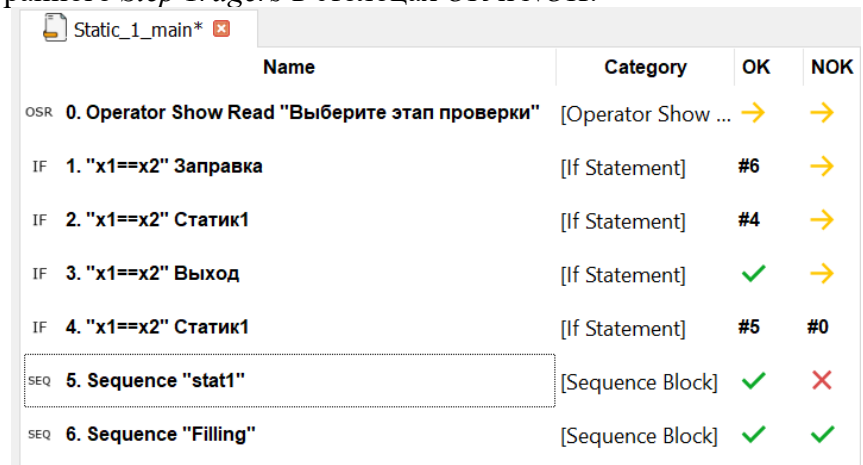
1. *Next Step* – переход к следующему шагу согласно тестовой последовательности;
2. *Break NOK* – завершение текущей тестовой последовательности с результатом *NOK*;
3. *Break OK* – завершение текущей тестовой последовательности с результатом *OK*;
4. Custom Step – переход выбирается разработчиком из выпадающего меню.

По умолчанию выбор следующий:



Из выпадающего списка можно выбрать нужный вариант.

Так же в секции формирования тестовых последовательностей присутствует отображение выбранного *Step Tragers* в столбцах *OK* и *NOK*:



- Если выбран *Next Step*, то пользователь видит желтую стрелочку вправо →.
- Если выбран *Break NOK*, то пользователь видит красный крестик ✕.

- Если выбран *Custom Step*, то пользователь видит номер примитива, к которому нужно перейти #5 .

10 Примитивы

Из пункта Секция выбора примитивов известно о нескольких категориях примитивов. В данном разделе приведено более подробное описание каждого примитива.

10.1 Basic

В данной категории представлен базовый набор примитивов, который позволяет печатать сообщения на принтере, запускать и останавливать таймер, устанавливать задержку по времени и др.

10.1.1 Sequence Block

Данный примитив позволяет включить в основную тестовую последовательность другую (заранее сформированную) тестовую последовательность. Поэтому для основной тестовой последовательности вложенная последовательность является примитивом. Соответственно для вложенной тестовой последовательности нужно определить входные параметры и выходные параметры (рисунок 9.1). Для определения входных и выходных параметров необходимо воспользоваться меню Edit Parameters кликнув правой кнопкой мыши по нужной последовательности в секции working. В разделе Input Parameters, по необходимости, добавить входные параметры. В разделе Output Parameters, по необходимости, добавить выходные параметры в зависимости от результата последовательности (in case OK, in case NOK). Выходные параметры подпрограммы могут быть не только целочисленные значения, но и промежуточные результаты самой подпрограммы, например целочисленный массив UDS ответа от ЭБУ.

10.1.1.1 Base Parameters (Sequence Block)

***Step* Targets**

Step Target принимает значение OK если **вложенная тестовая последовательность** была с Step Result true, и NOK – если Step Result false.

10.1.1.2 Specific Parameters

В данном окне необходимо из выпадающего списка выбрать тестовую последовательность, которая будет вызываться:



10.1.1.3 Пример использования (Sequence Block)

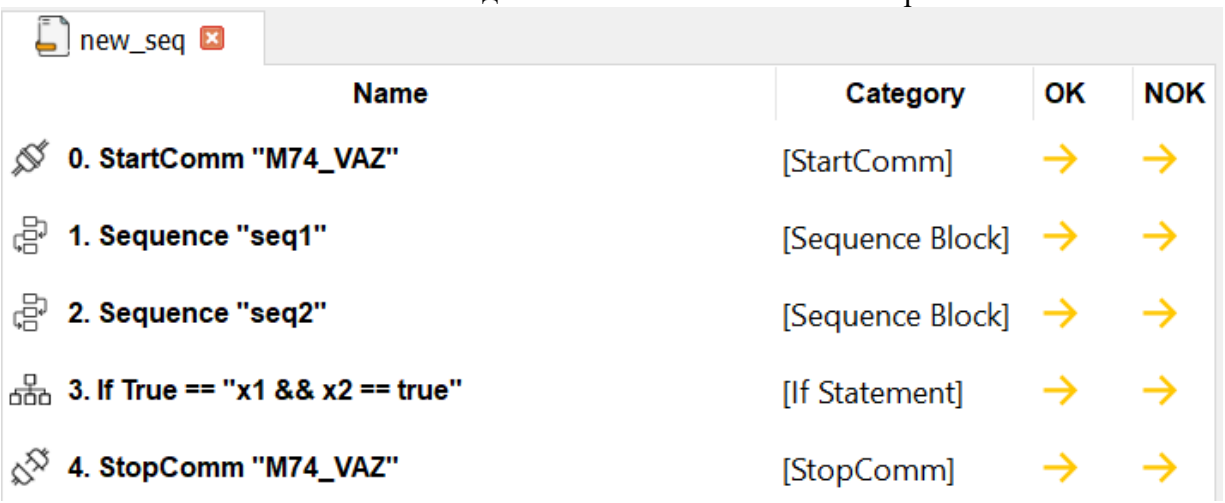
В качестве примера рассмотрим тестовую последовательность, состоящую из двух вложенных последовательностей (seq1 и seq2) и счетчика удачно выполненных подпрограмм.

Используемое оборудование для эксперимента:

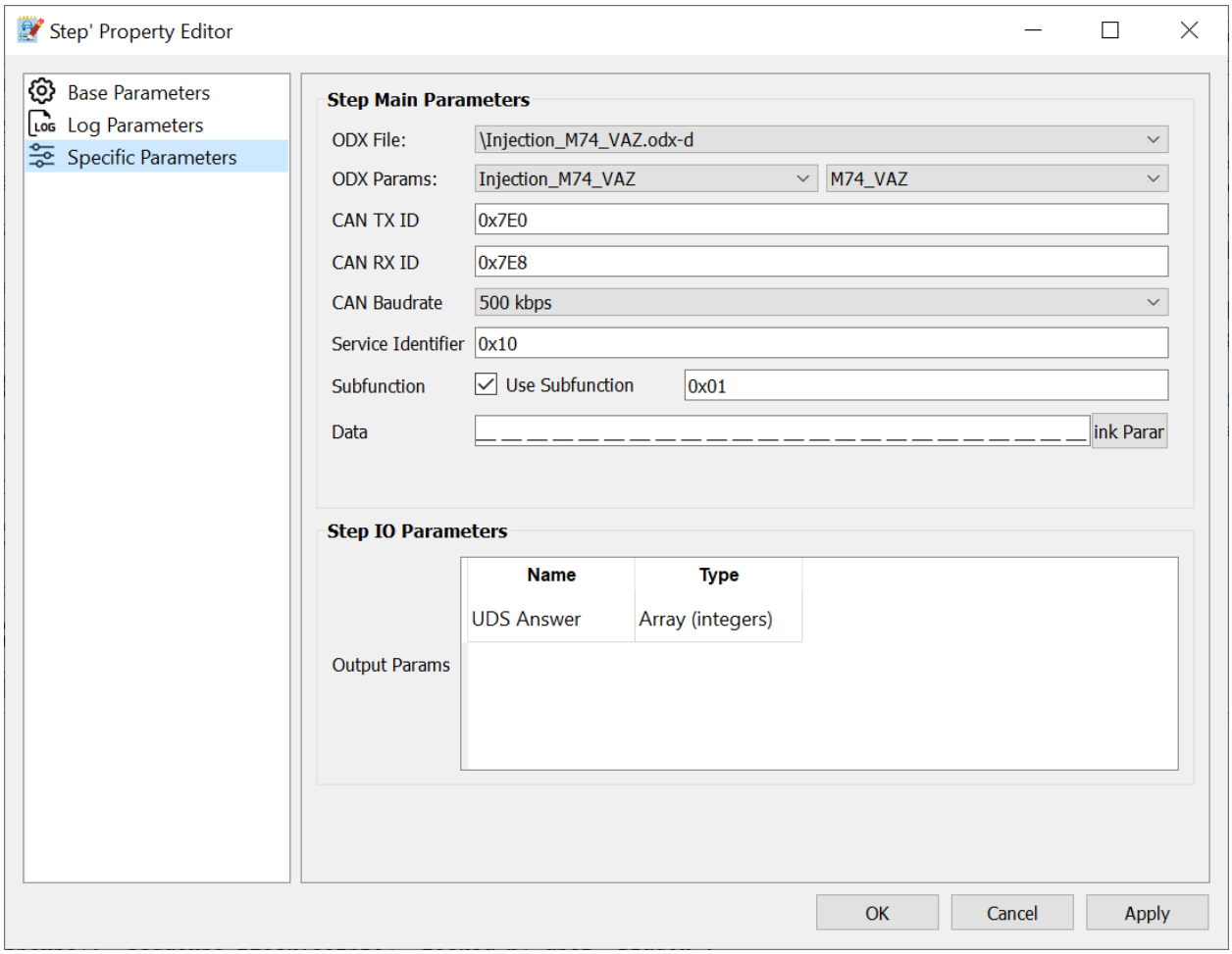
- ЭБУ – M74.9;
- MDA;
- ТСД;
- Ноутбук с установленным EOL Designer и TSE.

Также необходимо в папку ODX активного проекта загрузить odx-файл для ЭБУ M74.9.

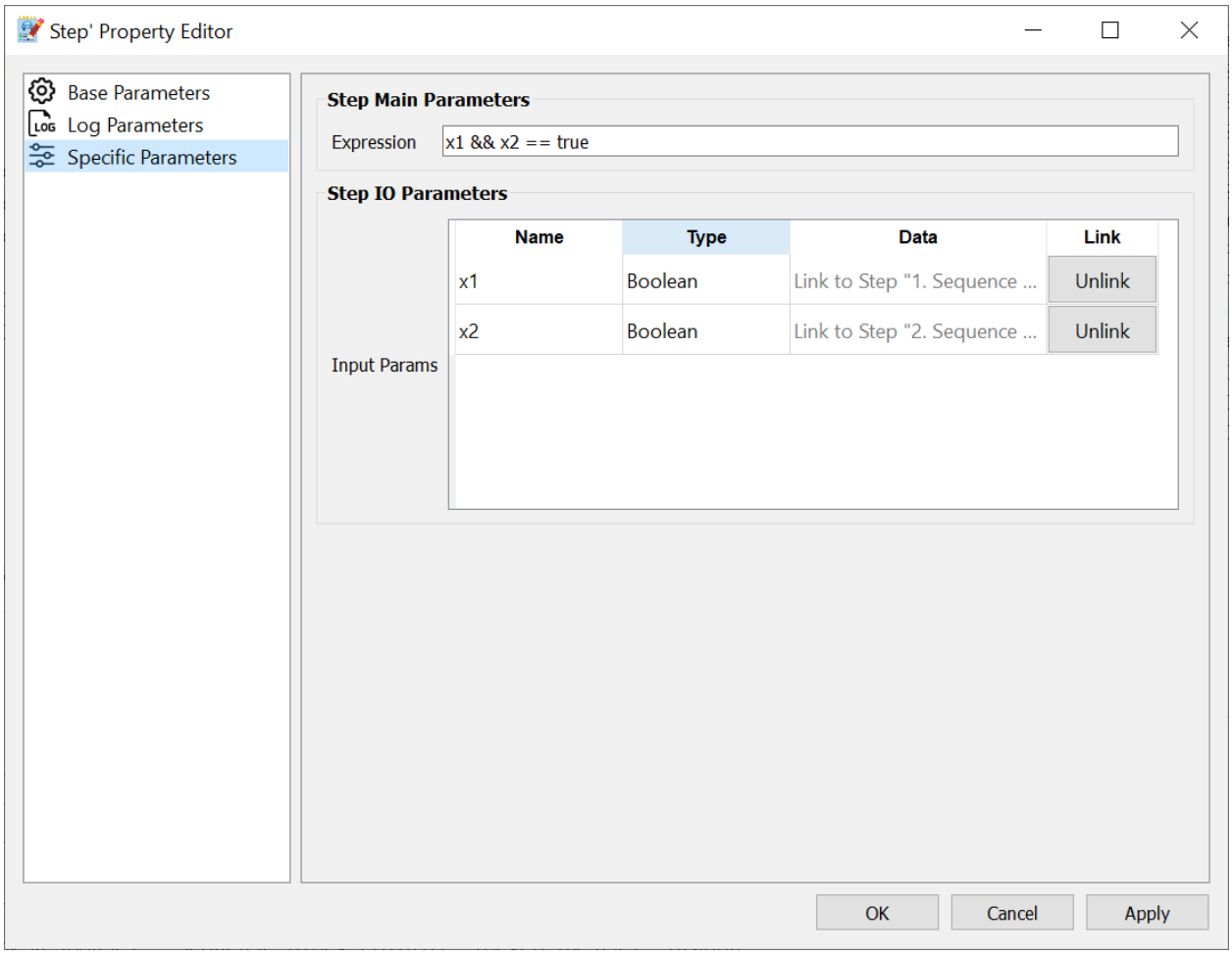
1. Основная тестовая последовательность состоит из четырех шагов:



2. Последовательности seq1 и seq2 идентичны и содержат примитив UDS Request с запросом 0x10 0x01:



3. Для анализа результатов выполнения подпрограмм используем примитив If Statement. Для этого в качестве значения переменных x1 и x2 нужно прилинковать Step Result вложенных последовательностей:



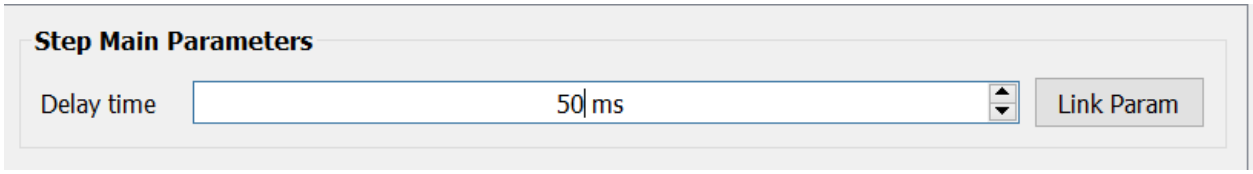
4. После сборки и запуска результат основной тестовой последовательности должен быть true.

10.1.2 Delay

Данный примитив позволяет осуществить задержку на заданное количество миллисекунд.

10.1.2.1 Specific Parameters (Delay)

В поле Delay time задается время задержки в миллисекундах:



10.1.2.2 Пример использования (Delay)

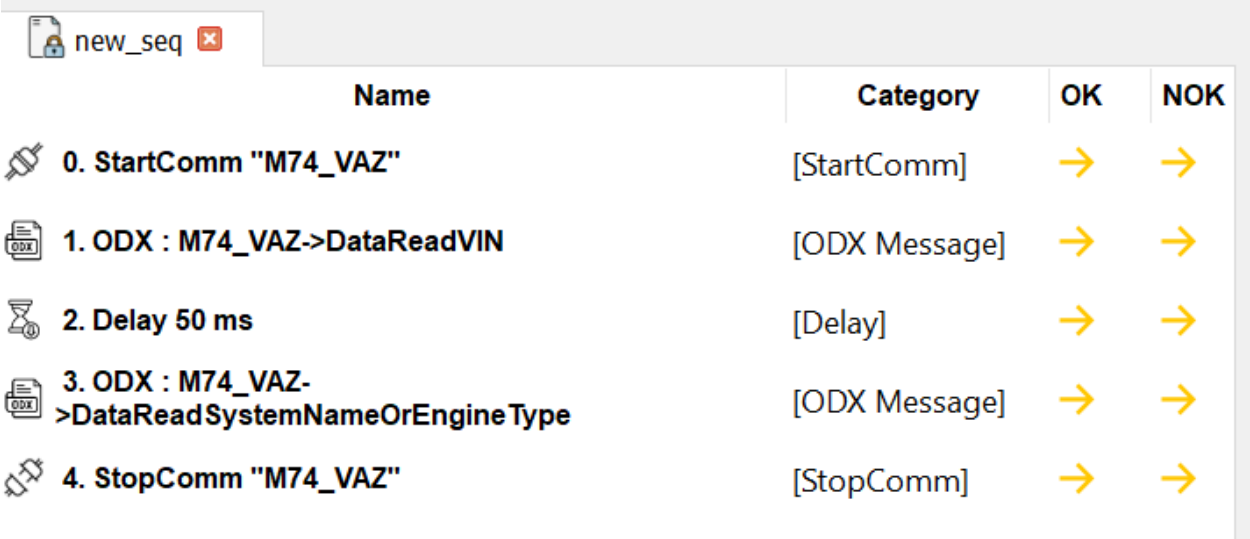
В качестве примера рассмотрим тестовую последовательность, в которой примитив Delay разделяет два UDS запроса.

Используемое оборудование для эксперимента:

- ЭБУ – M74.9;
- MDA;
- ТСД;
- Ноутбук с установленным EOL Designer и TSE.

Также необходимо в папку ODX активного проекта загрузить odx-файл для ЭБУ M74.9.

1. Основная тестовая последовательность состоит из четырех шагов:



2. После сборки и запуска результат тестовой последовательности должен быть true.

10.1.3 Print Log

Данный примитив позволяет отправить запрос принтеру на печать.

Формирование сообщения для печати

В каждом примитиве тестовой последовательности есть раздел параметров записи лога (Log Parmeters). Для того, чтобы на печать попала информация о необходимом примитиве нужно в разделе Log Parameters выбрать галочкой Printer для успешного выполнения примитива (in case OK) и/или неуспешного выполнения (in case NOK) и в соответствующем поле написать сообщение для каждого случая. Таким образом при вызове Print Log выведутся на печать результаты тех примитивов, которые выбрал разработчик.

Если примитив Print Log вызывается в тестовой последовательности несколько раз, то принтер будет печатать страницу столько раз, сколько вызывается данный примитив.

10.1.3.1 Specific Parameters (Print Log)

Step Main Parameters

В данном окне разработчик формирует печать лога для *in success case* и *in failure case*:



У разработчика тестовой последовательности есть возможность настройки печати лога.

Если в настройках Log parameters любого примитива выбран Printer in case OK и этот примитив был завершен с результатом OK, то Print Log напечатает сообщение из поля in success case.

Если в настройках Log parameters любого примитива выбран Printer in case NOK и этот примитив был завершен с результатом NOK, то Print Log напечатает сообщение из поля in failure case.

Если в настройках Log parameters любого примитива выбран Printer in case NOK и этот примитив был завершен с результатом OK, то Print Log напечатает сообщение из поля in success case.

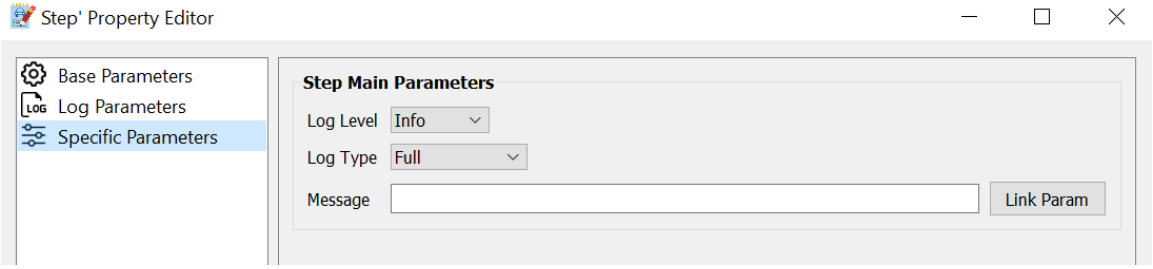
Если в настройках Log parameters любого примитива выбран Printer in case OK и этот примитив был завершен с результатом NOK, то Print Log напечатает сообщение из поля in success case, т.к. поле Printer in case NOK не было активировано.

10.1.4 Write Log

Данный примитив позволяет добавить информационную запись в выбранный тип и уровень лога тестовой последовательности.

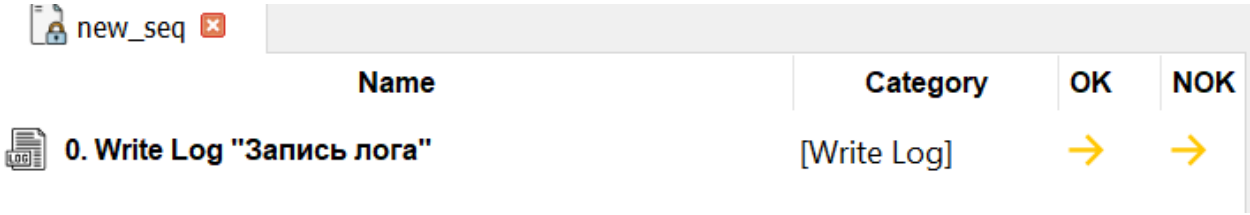
10.1.4.1 Specific Parameters (Write Log)

Разработчик имеет возможность выбрать уровень (Log Level) и тип лога (Log Type) из выпадающих списков. В поле *Message* ввести необходимое сообщение или с помощью клавиши *Link Param* выбрать доступное сообщение с любого другого примитива.

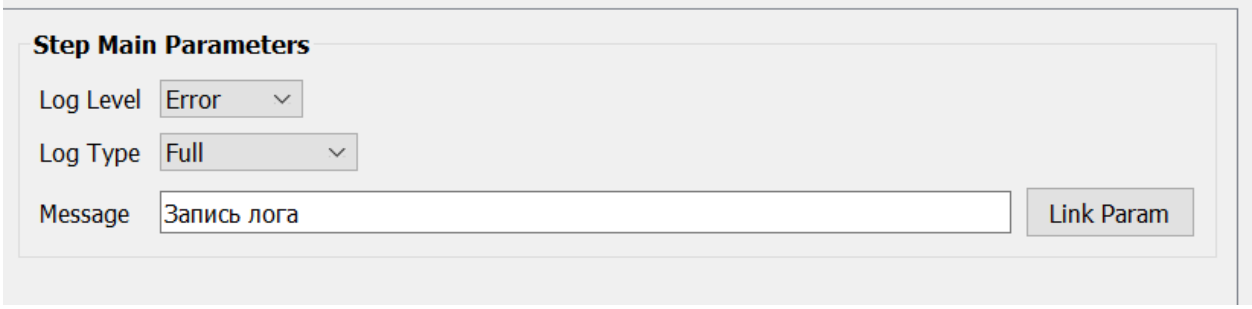


10.1.4.2 Пример использования (Write Log)

Рассмотрим тестовую последовательность, состоящую всего из одного примитива:



Specific Parameters примитива заполним следующим образом:



После сборки и запуска последовательности в приложении TSE во вкладке Sequence logs отображается лог:

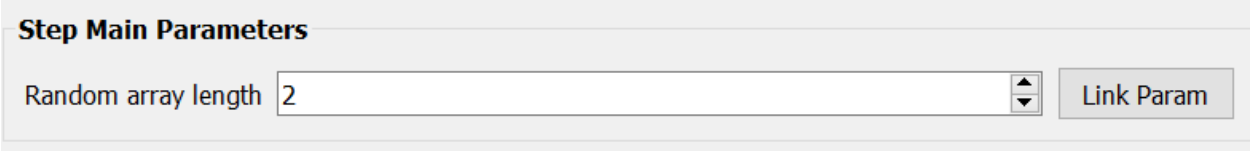


10.1.1 Random

Данный примитив генерирует массив данных типа integer.

10.1.1.1 Specific Parameters (Random)

В данном разделе задается длинна массива для генерации:



10.1.2 Timer Start

При вызове данного примитива запускается отсчет времени в мс. Если в тестовой последовательности был ранее вызван данный примитив, то предыдущий отсчет времени сбрасывается.

10.1.2.1 Specific Parameters (Timer Start)

В данном окне настройки отсутствуют.

10.1.3 Timer Check

При вызове данного примитива осуществляется фиксирование времени между Timer Start и Timer Check.

10.1.3.1 Specific Parameters (Timer Check)

В данном окне настройки выбираются автоматически.

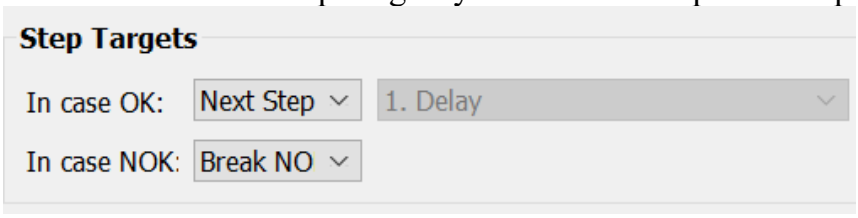
10.1.3.2 Пример использования Timer Start, Timer Check

Примитивы Timer Start, Timer Check, If Statement обычно используются вместе. В качестве примера составим последовательность, показанную на рисунке:

	Name	Category	OK	NOK
TS	0. Timer Start	[Timer Start]	→	✕
DY	1. Delay	[Delay]	→	✕
TCH	2. Timer Check	[Timer Check]	→	✕
IF	3. If Statement	[If Statement]	→	✕

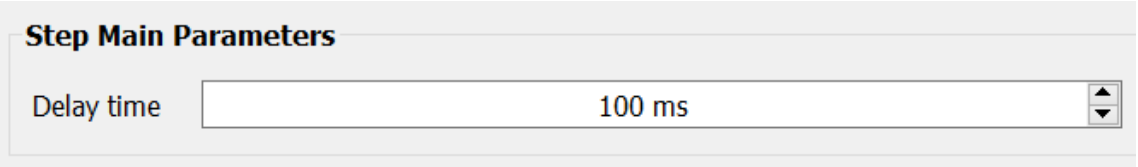
Для этого необходимо:

1. В Timer Start в Step Targets установить настройки Step Targets:



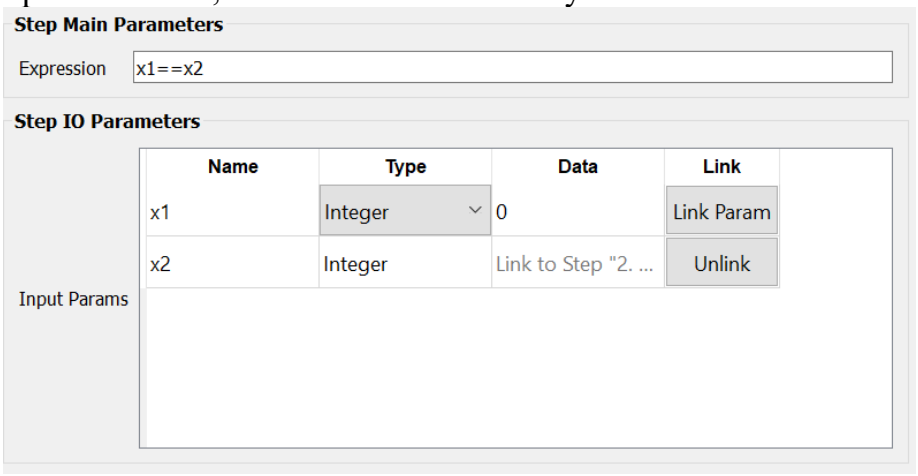
Остальные настройки можно не менять.

2. В Delay установить задержку 100 мс. Настройки Step Targets установить, как в Timer Start.

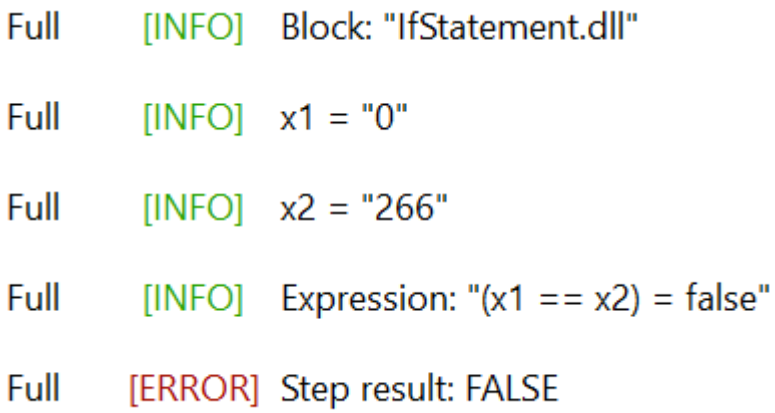


3. В Timer Check настройки Step Targets установить, как в Timer Start. Единицы измерения и тип данных выбираются системой автоматически.

4. В If Statement в окне Spesific Parameters написать равенство, например x1==x2. Дальше выбрать с помощью Link Param присваиваемые значения параметрам x1 и x2. В данном случает к параметру x1 ничего не прилинковано, т.к. Timer Start имеет нулевое значение.



5. После сборки и запуска последовательности в приложении TSE во вкладке Sequence logs отображается лог:

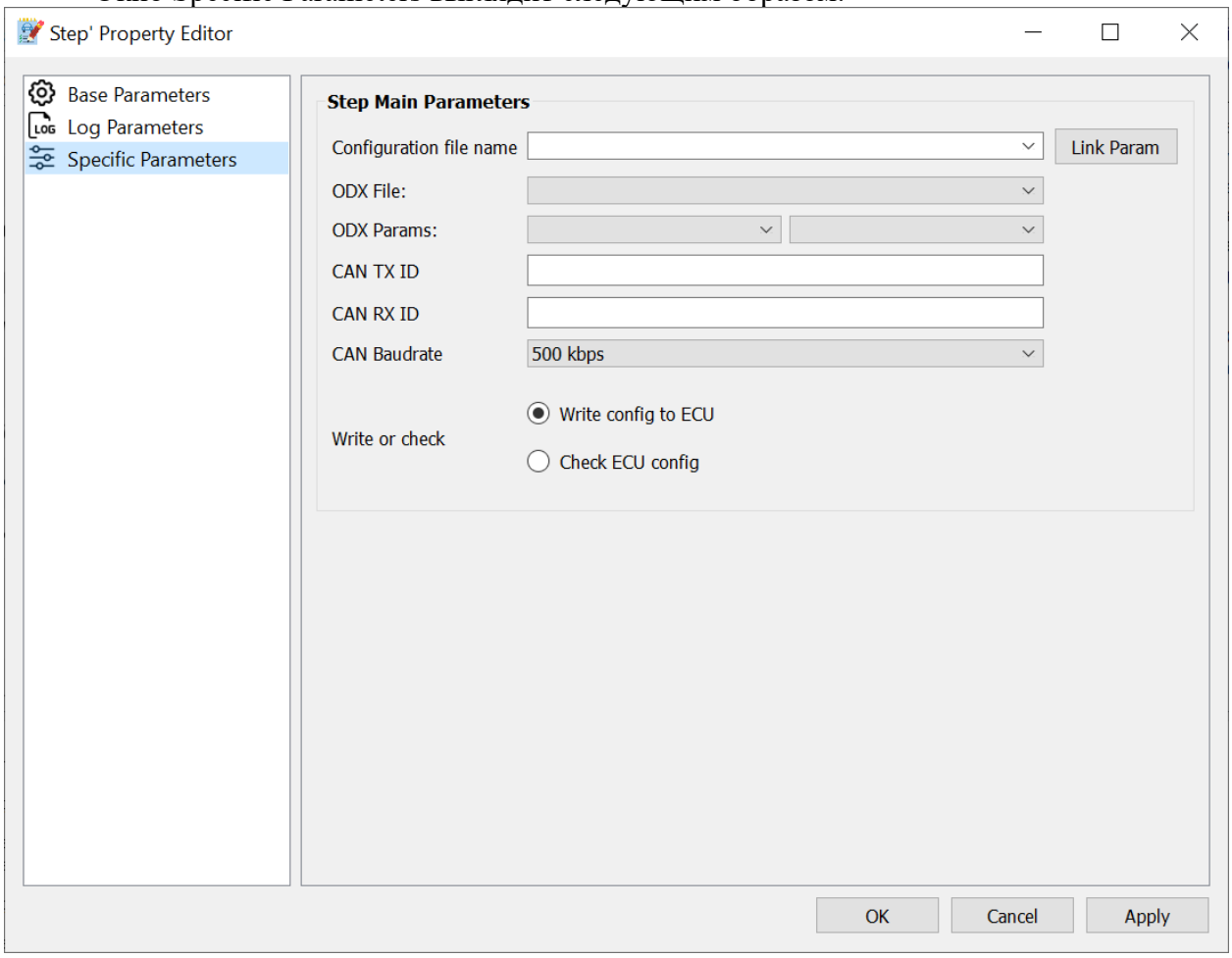


10.1.4 XML Configurator

Примитив XML Configurator предназначен для записи и проверки конфигурационных данных в выбранный ЭБУ.

10.1.4.1 Specific Parameters (XML Configurator)

Окно Specific Parameters выглядит следующим образом:



В поле Configuration File Name необходимо написать имя файла конфигурации или прилинковать его из другого примитива, например Read Mnemonic.

В полях CAN TX ID и CAN RX ID указывается идентификатор для взаимодействия тестового оборудования с ЭБУ.

В выпадающем списке CAN Baudrate выбрать скорость шины.

В поле Write or check необходимо выбрать тип процедуры. Write записывает конфигурационные данные в ЭБУ с помощью диагностического сервиса 0x2E. Check считывает конфигурационное данные из ЭБУ с помощью сервиса 0x22 и проверяет их на соответствие конфигурационным данным их XML файла.

10.1.5 Set Vehicle

Данный примитив предназначен для записи модели автомобиля в лог тестовой последовательности. Модель автомобиля используется для отображения и сортировки данных в Web Statistic.

10.1.5.1 Specific Parameters (Set Vehicle)

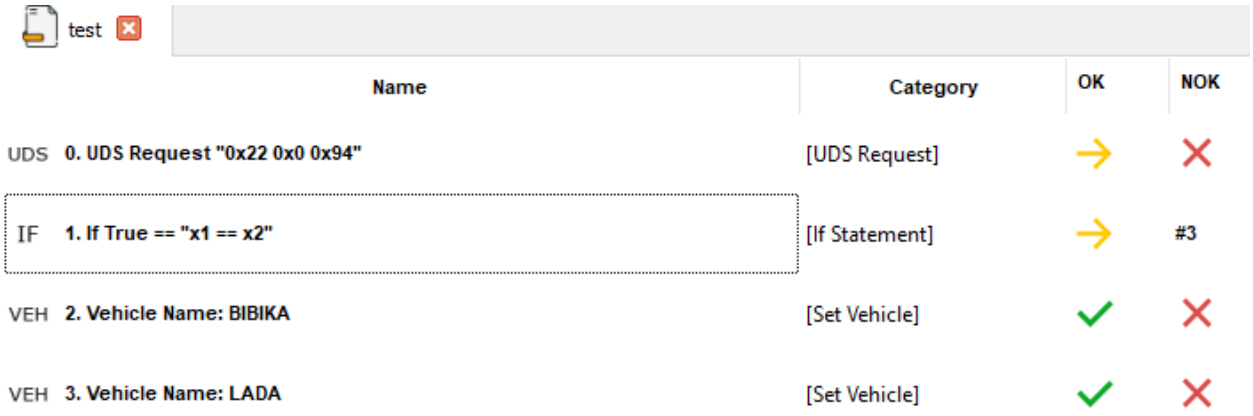
В поле Vehicle Name необходимо вписать модель автомобиля.



Если в тестовой последовательности Set Vehicle вызывается несколько раз, то система EOL запоминает имя только из последнего вызова примитива.

10.1.5.2 Пример использования (Set Vehicle)

Рассмотрим пример использования Set Vehicle на примере небольшой тестовой последовательности:



Если первый шаг тестовой последовательности true, то в качестве модели автомобиля LADA, если false – BIBIKA.

Записанный лог, если первый шаг тестовой последовательности true:

```
[INFO]    ------------------> Start Sequence "test" Step "UDS Request "0x22 0x0 0x94"" <------------------
[INFO]    Block: "UDSRequest.dll"
[INFO]    Sent: 0x22 0x00 0x94
[INFO]    Recv: 0x62 0x00 0x94 0x56 0x73 0x30 0x31 0x30 0x5F 0x31 0x30 0x32 0x00
[INFO]    Step result: TRUE
[INFO]    ------------------> Finish Sequence "test" Step "UDS Request "0x22 0x0 0x94"" <------------------

[INFO]    ------------------> Start Sequence "test" Step "If True == "x1 == x2"" <------------------
[INFO]    Block: "IfStatement.dll"
[INFO]    x1 == x2 = true
[INFO]    Step result: TRUE
[INFO]    ------------------> Finish Sequence "test" Step "If True == "x1 == x2"" <------------------

[INFO]    ------------------> Start Sequence "test" Step "Vehicle Name: LADA" <------------------
[INFO]    Block: "SetVehicle.dll"
[INFO]    Vehicle name setted to: "LADA"
[INFO]    Step result: TRUE
[INFO]    ------------------> Finish Sequence "test" Step "Vehicle Name: LADA" <------------------
```

Записанный лог, если первый шаг тестовой последовательности false:

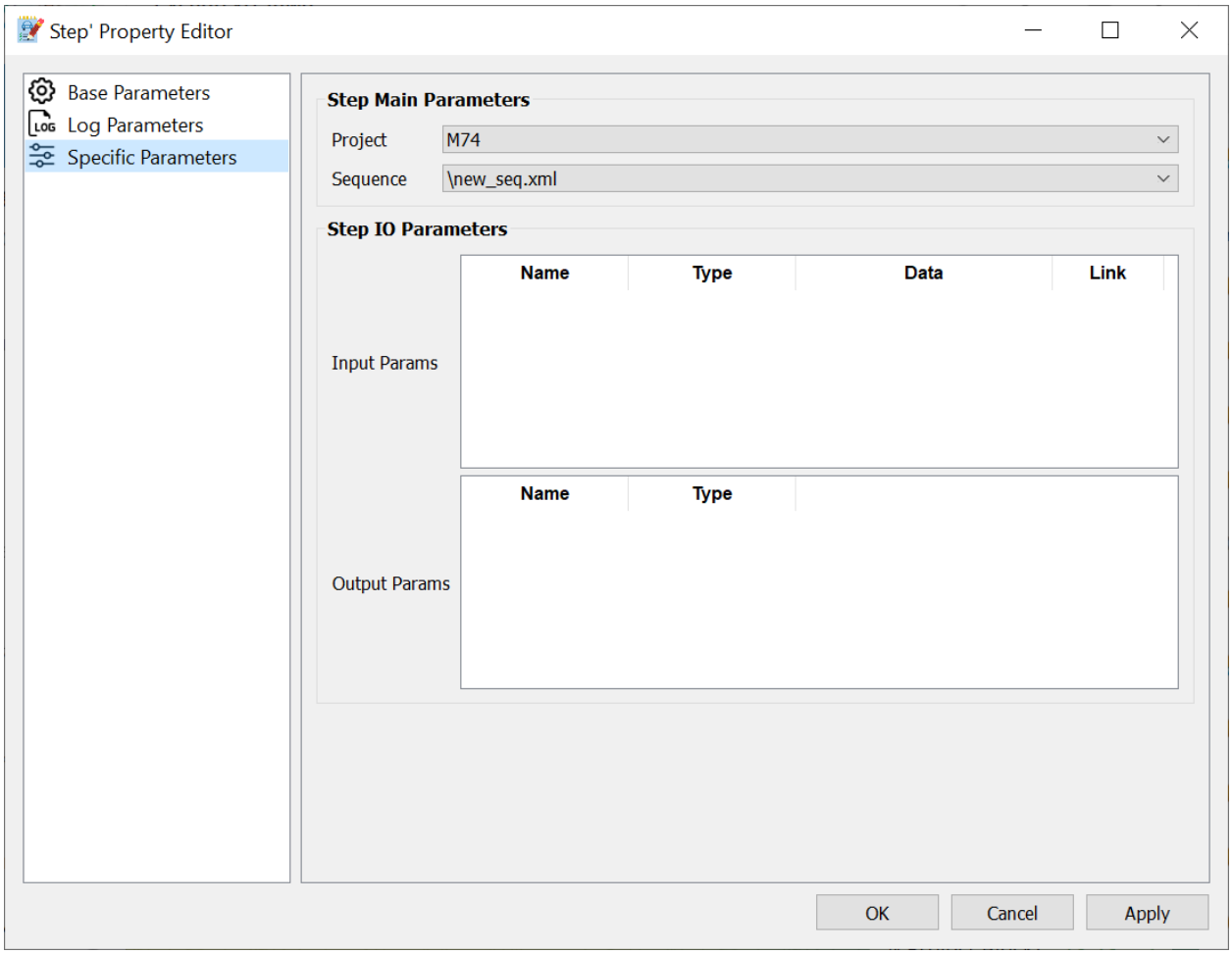
```
[INFO]    ------------------> Start Sequence "test" Step "UDS Request "0x22 0x0 0x94"" <------------------
[INFO]    Block: "UDSRequest.dll"
[INFO]    Sent: 0x22 0x00 0x94
[INFO]    Recv: 0x62 0x00 0x94 0x56 0x73 0x30 0x31 0x30 0x5F 0x31 0x30 0x32 0x00
[INFO]    Step result: TRUE
[INFO]    ------------------> Finish Sequence "test" Step "UDS Request "0x22 0x0 0x94"" <------------------

[INFO]    ------------------> Start Sequence "test" Step "If True == "x1 == x2"" <------------------
[INFO]    Block: "IfStatement.dll"
[INFO]    x1 == x2 = false
[ERROR]   Step result: FALSE
[INFO]    ------------------> Finish Sequence "test" Step "If True == "x1 == x2" <------------------

[INFO]    ------------------> Start Sequence "test" Step "Vehicle Name: BIBIKA" <------------------
[INFO]    Block: "SetVehicle.dll"
[INFO]    Vehicle name setted to: "BIBIKA"
[INFO]    Step result: TRUE
[INFO]    ------------------> Finish Sequence "test" Step "Vehicle Name: BIBIKA" <------------------
```

10.1.6 Project Block

Данный примитив позволяет в текущую тестовую последовательность включить другой проект или тестовые последовательности другого проекта. Важно, чтобы необходимый проект был открыт в секции проектов тестовых программ.

10.1.6.1 Base Parameters (Project Block)

Для включения тестовых последовательностей другого проекта в текущий необходимо из выпадающего списка *Project* выбрать проект. Затем из выпадающего списка *Sequence* выбрать необходимую тестовую последовательность.



10.1.7 Text Block

Данный примитив позволяет вводить информационный текст в секции формирования последовательности.

10.1.8 Get Section Name

Данный примитив возвращает имя участка, на котором воспроизводится тестовая последовательность.

Использование примитива рекомендовано для исключения ошибки, когда на чек-станции определенного участка была ошибочно загружена не соответствующая ей тестовая последовательность.

Также данный примитив рекомендован к использованию для участков ремонтной зоны, т.к. позволяет ветвить тестовую последовательность в зависимости от нужного участка.

10.1.9 Printer Log Start

Данный примитив предназначен для обнуления (стирания) сформированного сообщения для печати.

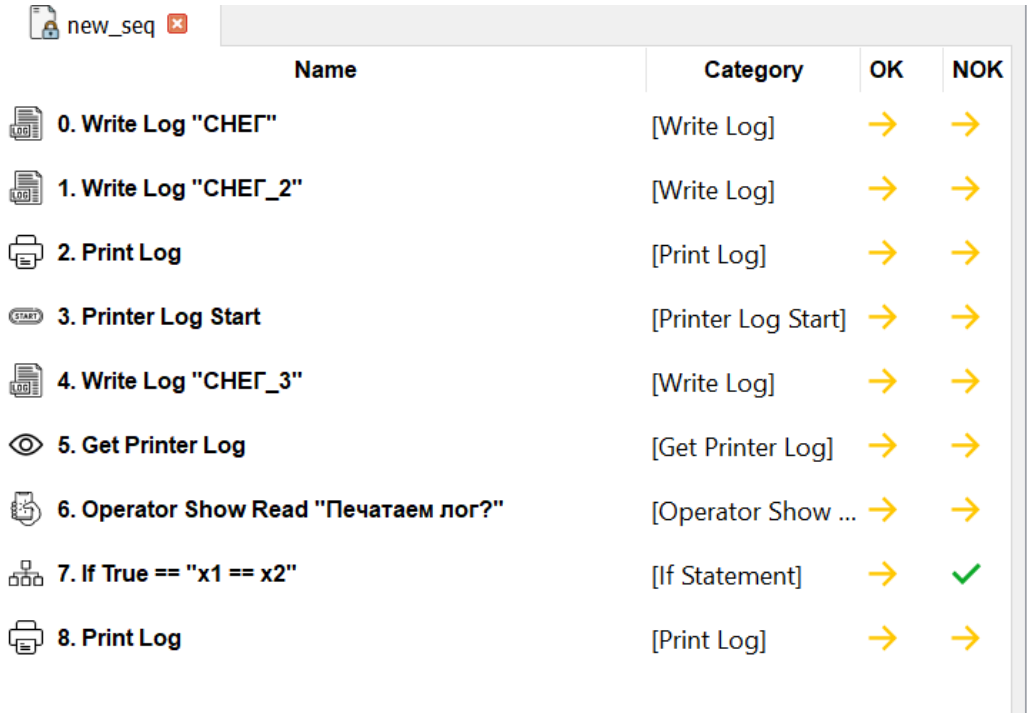
Формирование сообщения для печати осуществляется в каждом примитиве тестовой последовательности в разделе *Log Parameters*.

Входные/выходные параметры	Название	Тип
Input Parameters	Отсутствуют	
Output Parameters	Отсутствуют	
Step Result	Таблица 9.1	

10.1.9.1 Пример использования (Printer Log Start)

В представленной ниже тестовой последовательности *Printer Log Start* располагается на третьей позиции.

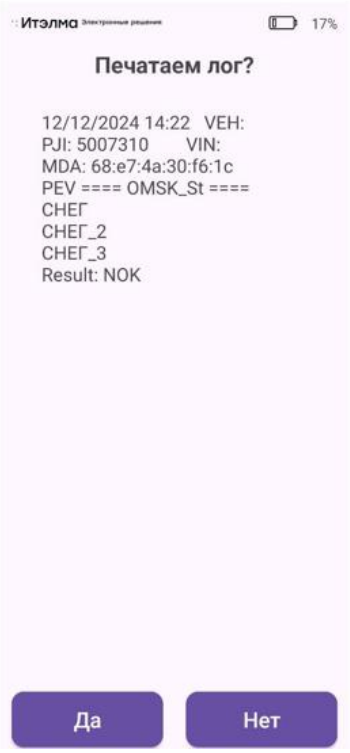
Для наглядного объяснения принципа работы *Printer Log Start* в начале тестовой последовательности расположены два подряд примитива *Write Log*. Далее расположен примитив *Print Log*, который отправляет на печать «СНЕГ» и «СНЕГ_2». Затем расположен *Printer Log Start* и за им следует набор примитивов включая отображения лога на экране ТСД и запроса у оператора печати лога на принтере.



При наличии в последовательности примитива *Printer Log Start* для печати формируется следующее лог:



При отсутствии в последовательности примитива *Printer Log Start* для печати формируется следующее лог:



10.1.10 Get Printer Log

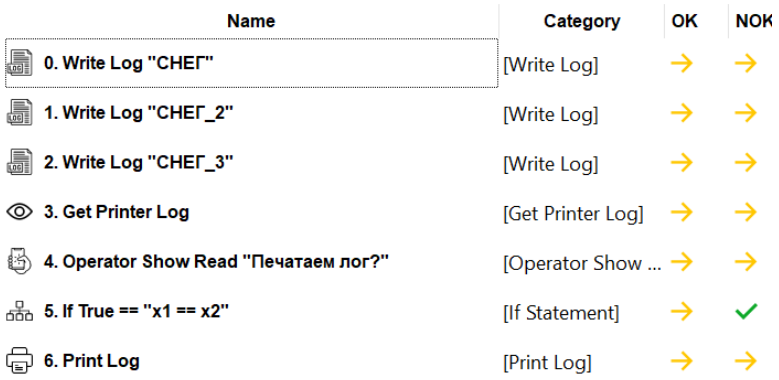
Данный примитив позволяет отобразить сформировавшийся лог для принтера на экране ТСД.

Входные/выходные параметры	Название	Тип
Input Parameters	Отсутствуют	
Output Parameters	Printer Log	String
Step Result	Таблица 9.1	

Использование Get Printer Log рекомендовано с примитивом Operator Show Read.

10.1.10.1 Пример использования (Get Printer Log)

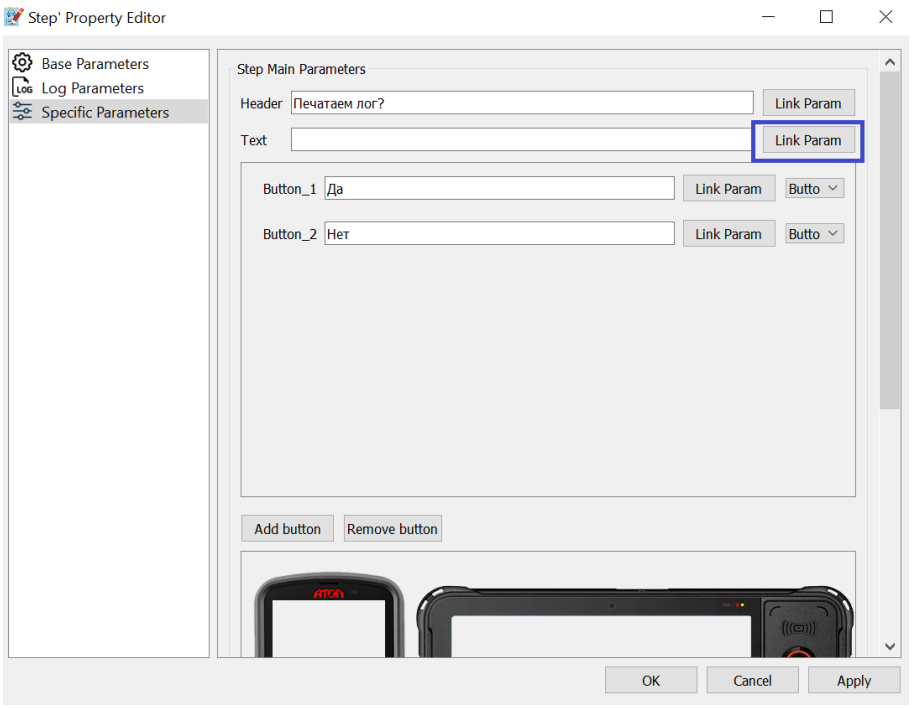
На рисунке ниже представлена тестовая последовательность с использованием Get Printer Log:



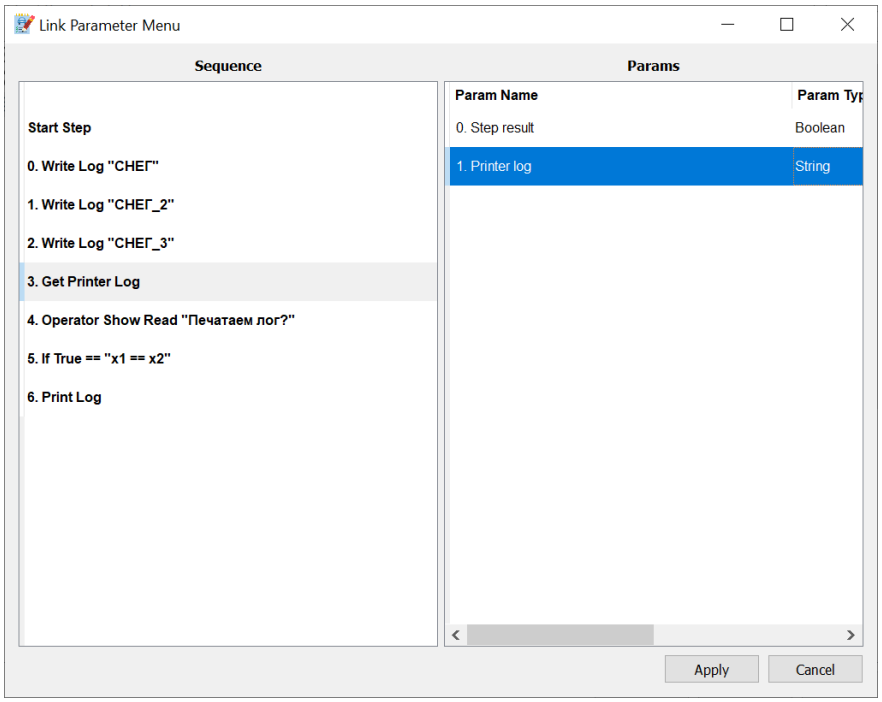
Первые три примитива *WriteLog* формируют лог. Далее *Get Printer Log* в паре с *Operator Show Read* выводят лог на экран ТСД.

Для связи *Get Printer Log* и *Operator Show Read* нужно в свойствах *Operator Show Read* в качестве текса для вывода сообщения выбрать выходной параметр примитива Get Print Log. Последовательность действий показана на скриншотах ниже:

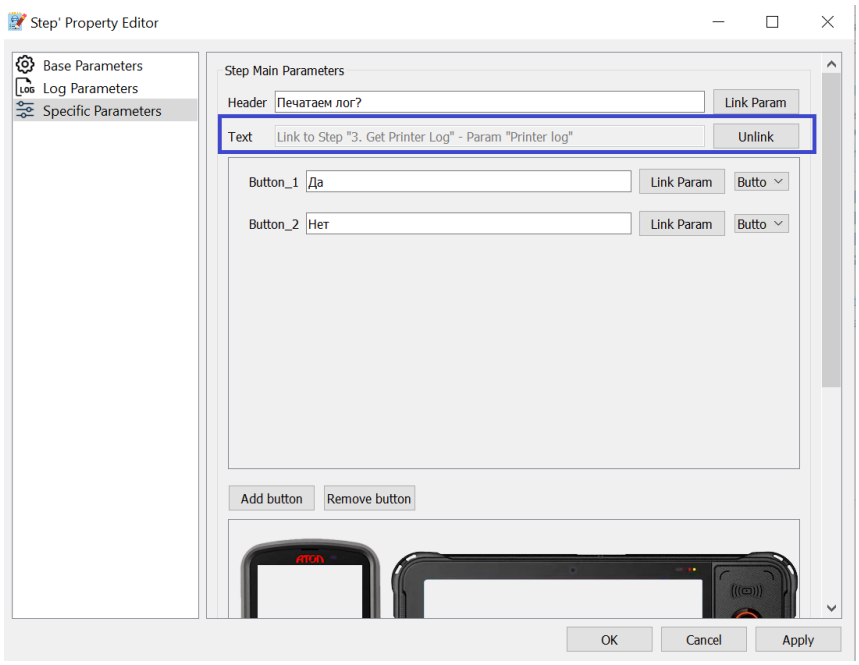
1. Свойства *Operator Show Read*



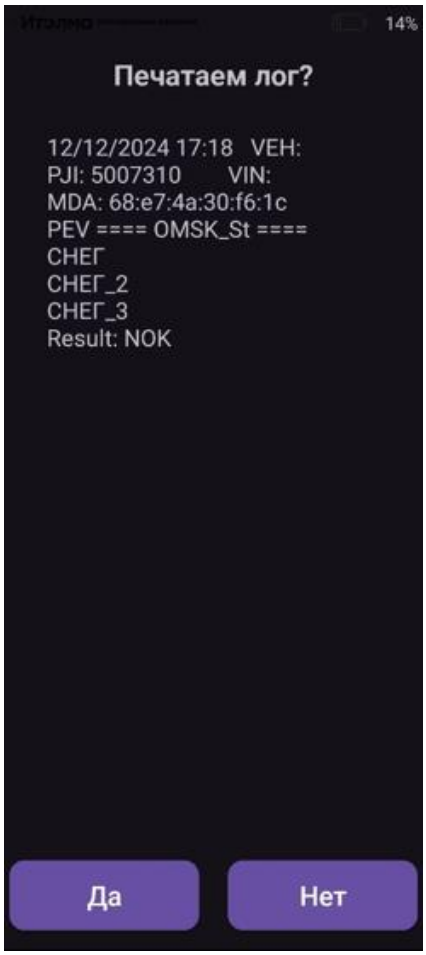
2. Линковка выходного параметра *Get Log Printer* и входного параметра *Operator Show Read.*



3. Свойства *Operator Show Read* после процедуры *Link.*



Отображаемый лог данной последовательности:



10.1.1 Get Seed

Данный примитив возвращает случайное число заданного размера в количествах бит.

10.2 Bench

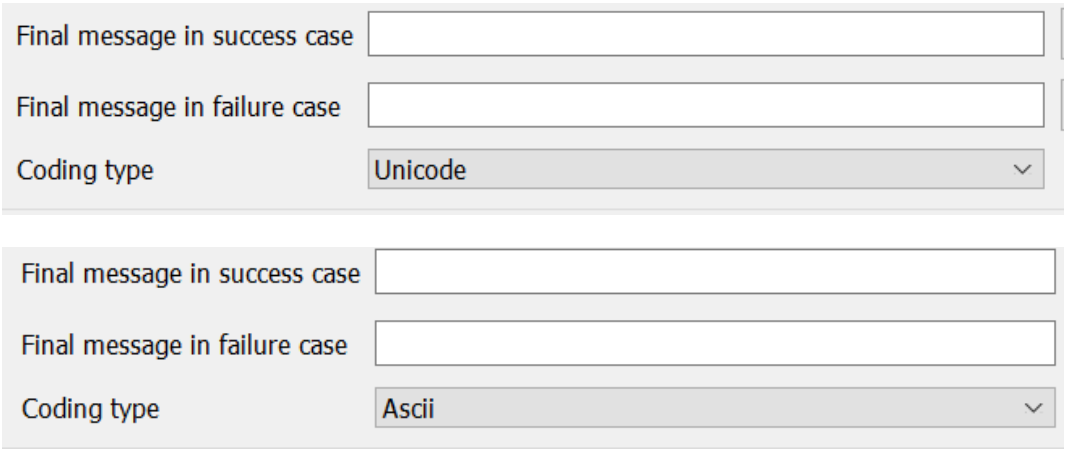
Данный раздел примитивов предназначен для взаимодействия системы EOL с испытательными стендами такими как стенд установки углов колес, роликовый стенд, стенд заправки тормозной жидкостью.

10.2.1 Bench Report

Bench Report, как правило, сообщает стенду о завершении цикла коммуникации и содержит информацию для печати на принтере.

10.2.1.1 Specific Parameters (Bench Report)

Сообщение для печати может быть отправлено с использованием кодировок Unicode и Ascii:

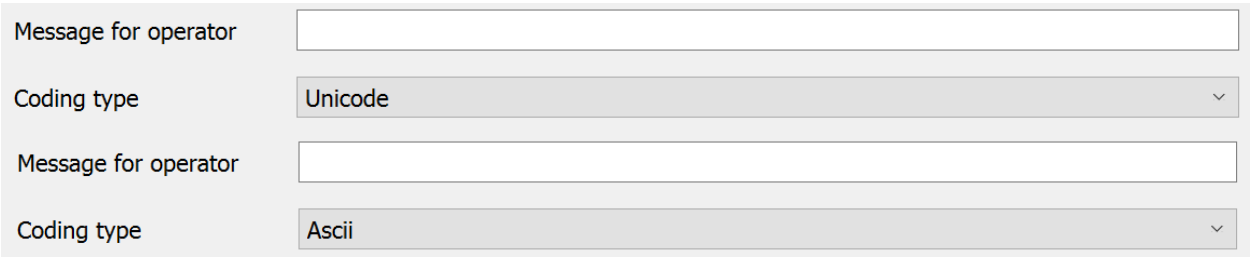


10.2.2 Bench Show Message

Данный примитив предназначен для отправки сообщений оператору.

10.2.2.1 Specific Parameters (Bench Show Message)

Сообщение для оператора может быть отправлено с использованием кодировок Unicode и Ascii:



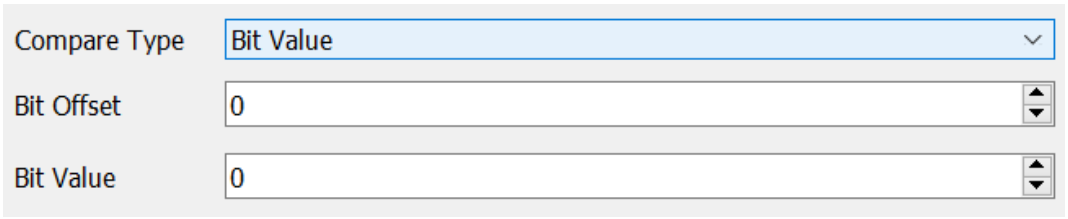
10.2.3 Wait Synchro Word

Данный примитив выполняет функцию ожидания синхроимпульса от стенда по протоколу Modbus.

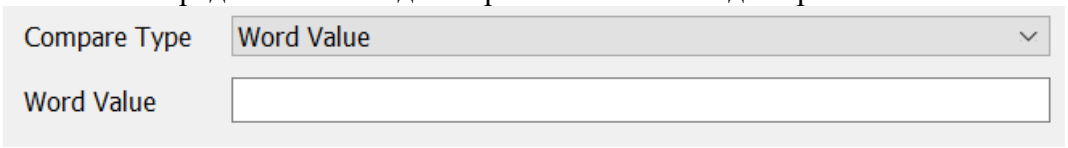
10.2.3.1 Specific Parameters (Wait Synchro Word)

Synchro Number – номер синхроимпульса.
Timeout – время ожидания синхроимпульса в секундах.
Compare Type – тип сравнения. В данном случае ожидаемый синхроимпульс сравнивается со значением в виде: Bit Value, Word Value, Word Limit.

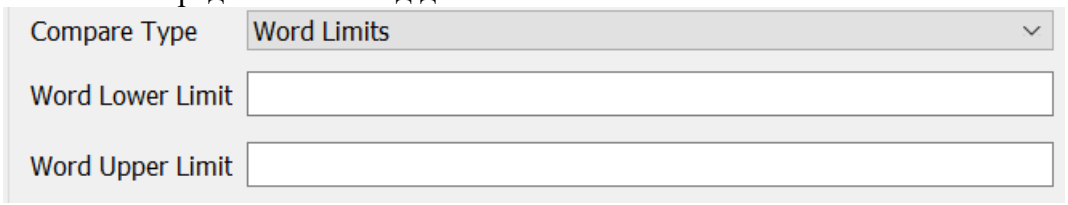
Bit Value предполагает настройку битового смещения Bit Offset и соответственно его значения Bit Value.



Word Value предполагает ввод конкретного значения для сравнения:



Word Limit предполагает ввод диапазона:



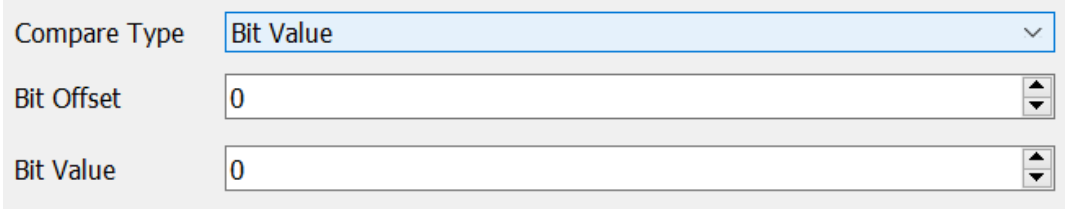
Система EOL ожидает в течении времени (Timeout) синхроимпульс и сравнивает его с ожидаемым значением. Результатом выполнения примитива является успешное или неуспешное получение синхроимпульса. Если система EOL в течении Timeout ничего не получает, то генерируется ошибка.

10.2.4 Write Synchro Word

Данный примитив выполняет функцию отправки синхроимпульса от системы EOL к стенду по протоколу Modbus.

10.2.4.1 Specific Parameters (Write Synchro Word)

Synchro Number – Номер синхроимпульса.
Timeout – время ожидания синхроимпульса в секундах.
Compare Type – тип сравнения. В данном случае ожидаемый синхроимпульс сравнивается со значением в виде: Bit Value, Word Value, Word Limit.
Bit Value предполагает настройку битового смещения Bit Offset и соответственно его значения Bit Value.



Word Value предполагает ввод конкретного значения для сравнения:



10.3 Current Clamp

Для получения данных о значениях потребляемой мощности и силы тока электропотребителей на борту автомобиля, таких как: датчики, актуаторы, помпы, лампочки и др. используются примитивы работы с токовыми клещами.

Беспроводные токовые клещи передают значения замеров силы тока на чек-станцию системы EOL.

В графической среде разработки тестовых последовательностей «Itelma EOL Designer» для работы с токовыми клещами предусмотрены примитивы *Current Clamp Connect*, Current *Clamp Start* и *Current Clamp Stop* (рисунок 10.1).

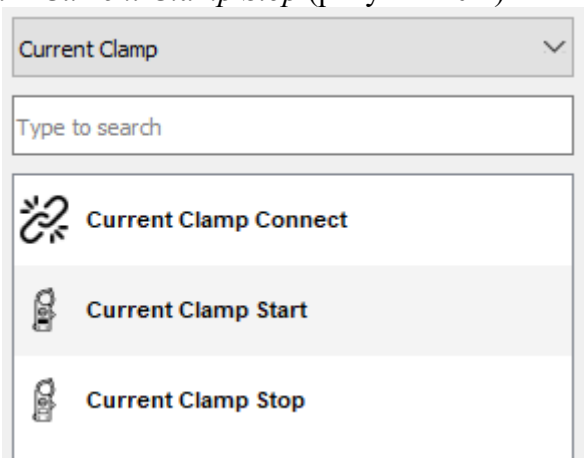


Рисунок 10.1 - Представление примитивов работы с токовыми клещами в области выбора примитивов.

Цикл измерения – это замер тока от Current Clamp Start до Current Clamp Stop.

Характеристики токовых клещей:

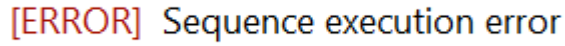
- Частота обновления сигнала с токовых клещей составляет 10 [мс].

10.3.1 Current Clamp Connect

Данный примитив предназначен для установления связи с токовыми клещами в соответствии с настройками в Web Statistic.

В Web Statistic каждые токовые клещи сопряжены со своей чек-станцией и имеют свой MAC и IP адреса.

Если связь с токовыми клещами установить не удалось, то воспроизведение тестовой последовательности прекращается с ошибкой:



10.3.2 Current Clamp Start

Данный примитив запускает процедуру замера токовыми клещами. *Current Clamp Start* должен быть установлен и настроен в тестовой последовательности до того, как будет активирована измеряемая электрическая нагрузка. Примитив позволяет совершить только один цикл замера, который обязательно должен быть завершен примитивом *Current Clamp Stop*.

10.3.2.1 Specific Parameters (Current Clamp Start)

В данном окне представлены параметры, настраиваемые разработчиком для установки режима измерения и временных параметров токовый клещей для проведение успешного замера тока (рисунок 10.2), а также приведен пример кривой токового замера, на котором отображены контрольные точки настраиваемых параметров (рисунок 10.3).

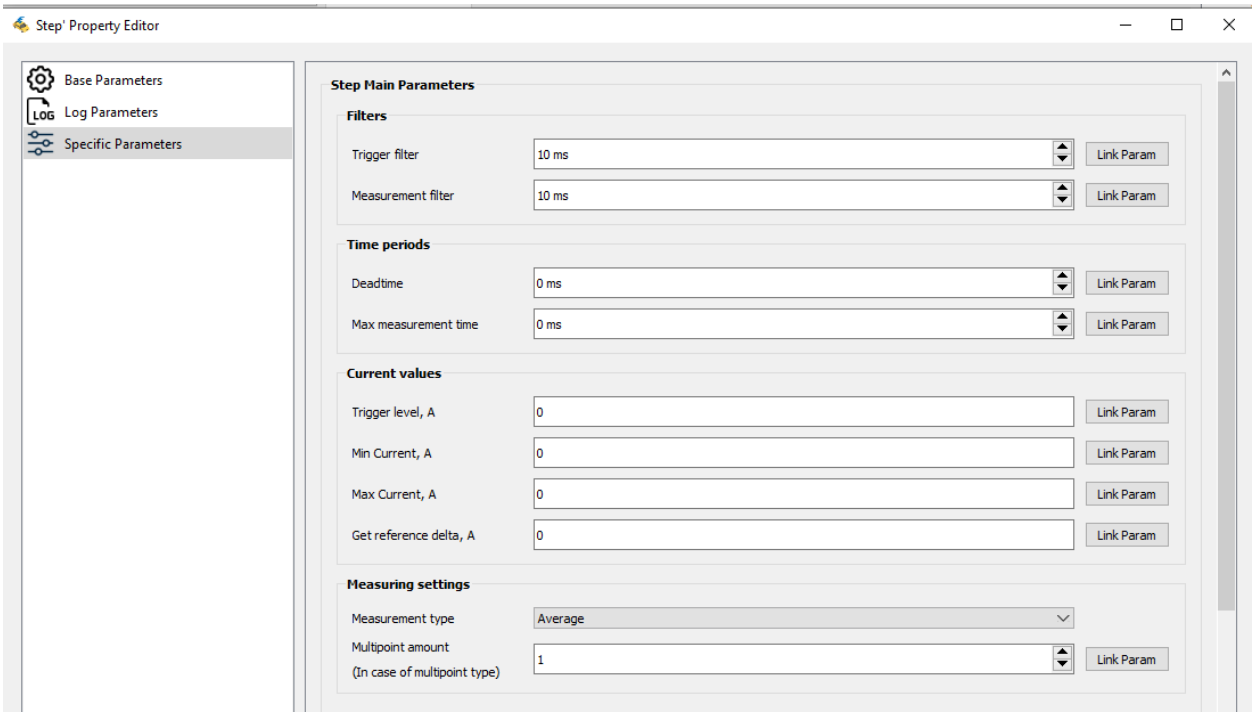


Рисунок 10.2 – Настраиваемые параметры для работы с токовыми клещами.

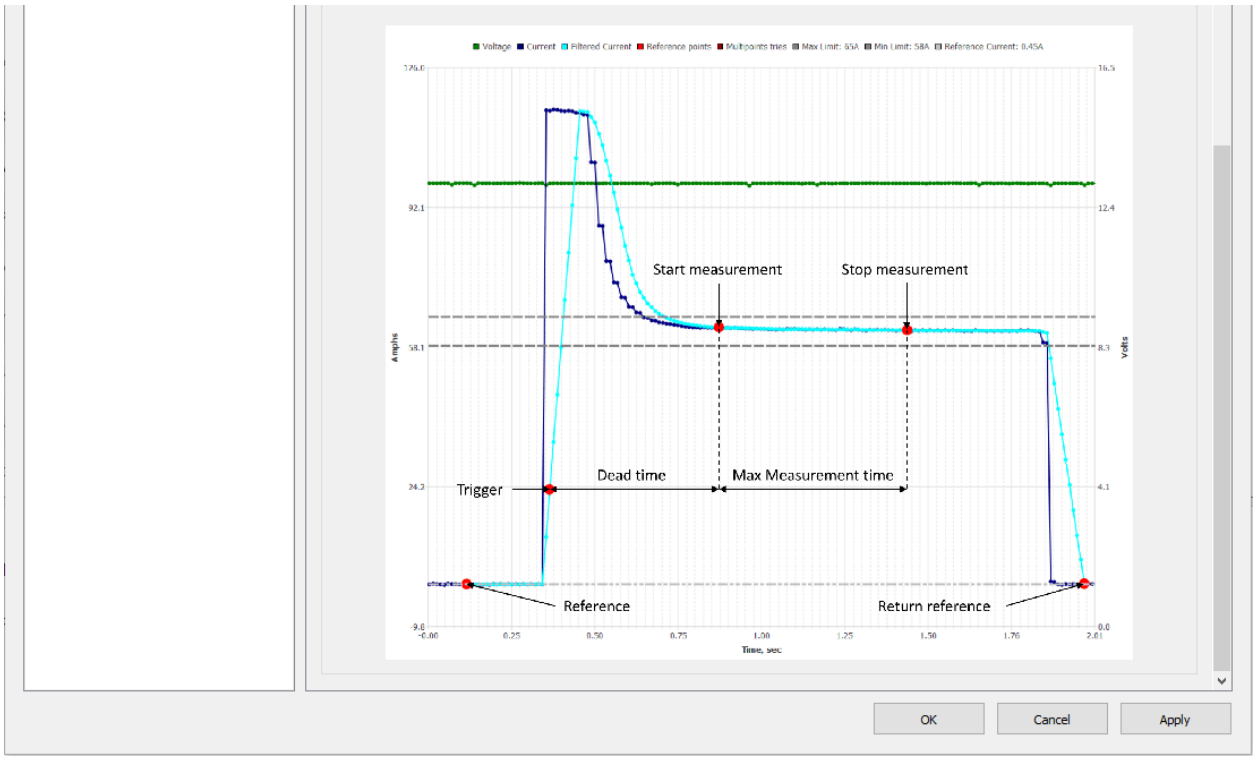


Рисунок 10.2 – Графическое изображения токового замера с указанием временных и количественных параметров.

Поле Filters включает в себя настройку фильтрации выходного сигнала с токовых клещей. Для фильтрации используется принцип скользящего среднего. Данный фильтр считывает значения последних точек замера за указанный интервал окна (см. рисунок 10.3) и усредняет их для получения выходных данных.

Примитив работы с токовыми клещами имеет возможность настройки двух фильтров среднего скользящего:

- Trigger filter – фильтр действует от точки «Reference» до точки срабатывания по триггеру «Trigger» (см. рисунок 10.2);
- Measurement filter - фильтр применяется от старта таймера «Dead time» и до окончания таймера «Measurement time» (см. рисунок 10.2).

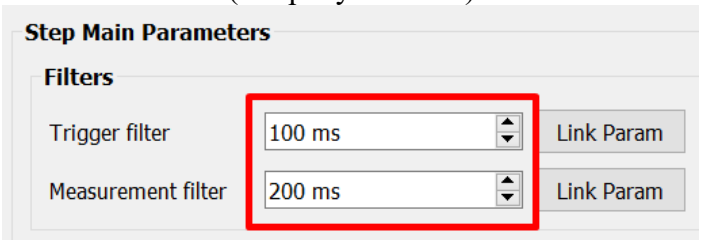


Рисунок 10.3 - Область задания интервала окна фильтра скользящего среднего в миллисекундах.

В точке «Reference» графика (см. рисунок 10.2) производится замер тока (по умолчанию) относительного которого вычисляется значение тока нагрузки за время «Measurement time». Отображение этой точки на кривой тока для пользователя несет информационный характер.

Поле Time Period включает в себя настраиваемые характеристики такие как «Deadtime» и «Measurement time».

«Dead time» – время стабилизации тока с момента достижения порога срабатывания «Trigger level». Это время выделяется на фазу переходного процесса, после активации нагрузки.

«Measurement time» – время измерения установившегося значения тока, наступает после завершения времени «Dead time».

Поле Current values включает в себя настраиваемые характеристики такие как «Trigger level» и «Min Current», «Max Current», «Get reference delta».

«Trigger level» – порог срабатывания в амперах [A]. Этот параметр при достижении указанного значения силы тока определяет старт таймера «Dead time» и обозначает активацию соответствующего электропотребителя.

«Min Current» и «Max Current» – верхний и нижний лимиты в амперах [A] допустимых значений измеряемого тока.

«Get reference delta» – допустимое значение разницы токов «Reference» и «Return reference», взятых по модулю.

Поле Measuring settings определяет настройку типа измерения силы тока. Применяется два способа для определения установившегося значения силы тока за время «Measurement time», которые разработчик может выбрать в опции «Measurement type»:

- Average – значение тока нагрузки вычисляется по среднему значению всех измеренных точек. Полученное значение тока является корректным, если не выходит за границы диапазона (*Min Current…Max Current*). В случае возникновения ошибки на тикет и в web-statistic выводится значение, соответствующее последней точке замера.

- Multipoint - многоточечное измерение тока нагрузки. Полученное значение тока является корректным, если в диапазоне (*Min Current…Max Current*) детектируется как минимум «n» последовательных точек.

При выборе типа измерения Multipoint разработчику становится доступным задать количество «n» последовательных точек в параметре «Multipoint amount (in case of multipoint type)».

Примечание: значение времени, полученное перемножением *Multipoint amount*(10 мс)* не должно превышать время, указанное в параметре «*Measurement time».*

10.3.3 Current Clamp Stop

Данный примитив останавливает процедуру замера токовыми клещами. *Current Clamp Stop* в тестовой последовательности должен располагаться после примитива *Current Clamp Start*.

10.3.3.1 Specific Parameters (Current Clamp Stop)

В данном окне представлена возможность настройка параметра «Wait return reference value» (рисунок 10.4).

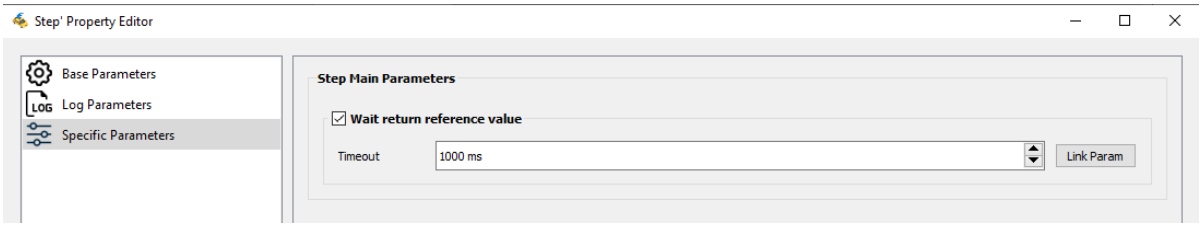


Рисунок 10.4 - Окно настройки параметров для работы с токовыми клещами

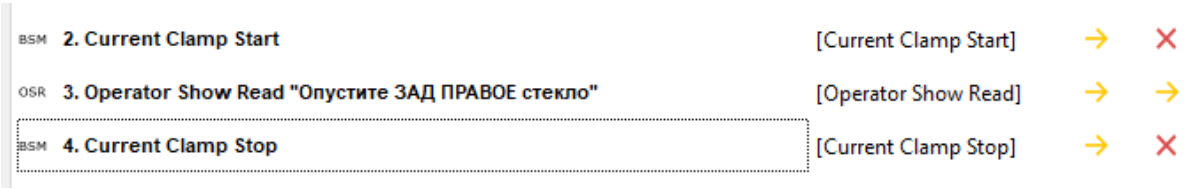
Wait return reference value – условие завершения замера токовыми клещами, при котором ожидается установка уровня сигнала в точке «*Return reference*», равного «*Reference*» *(+/-) «Get reference delta», [A]* за время «*Timeout*» (см. рисунок 10.2). Если по истечении таймера соответствующий уровень сигнала не был установлен, то цикл работы с токовыми клещами завершается с ошибкой.

Переход в «*Return reference*» осуществляется отключением ранее активированной нагрузки (возможно как диагностикой IO Control $2F, так и ручным отключением оператором на линии).

Если опция «*Wait return reference value*» не активирована, то завершение цикла работы токовых клещей произойдет по истечении времени «*Measurement time*».

Timeout – время ожидания корректного значения «*Return reference*», если выбрана опция «*Wait return reference value*».

10.3.3.2 Примеры построения тестовых последовательностей с примитивами Current Clamp Start и Current Clamp Stop





10.4 DEV File

Данный вид примитивов предназначен для работы с DEV файлом.

Каждый автомобиль имеет свой уникальный номер PJI. Каждому PJI соответствует определенный DEV файл. DEV файл содержит конфигурационные параметры автомобиля, критерии и мнемоники.

10.4.1 Mnemonic Read

Данный примитив предназначен для считывания значения заданной мнемоники и проверки наличия критерия.

10.4.1.1 Specific parameters (Mnemonic Read)

В данном окне в поле *Mnemonic Name* необходимо написать (или прилинковать с помощью кнопки Link) имя мнемоники или критерия.

В поле *Output Params* отображено имя и тип выходного параметра примитива при считывании значения мнемоники.

Если запрашивается мнемоника, то выходным параметром примитива будет значение мнемоники и соответственно результат примитива будет true. Если запрашиваемая мнемоника отсутствует, то результат примитива будет false.

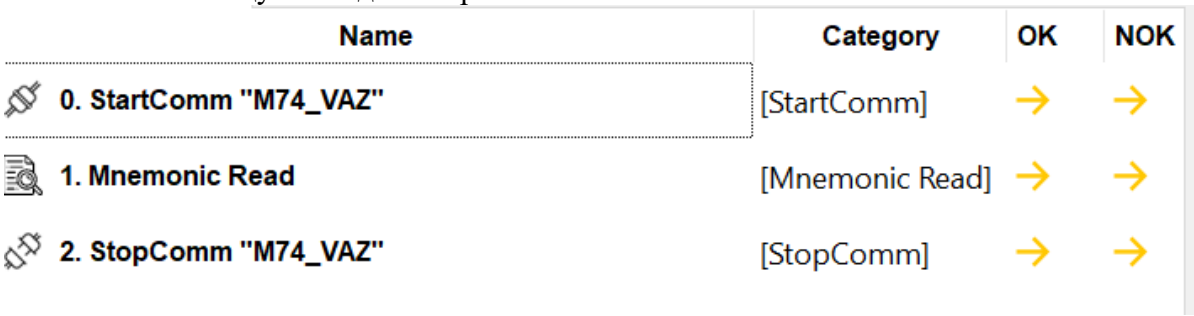
Если запрашивается критерий, то результатом примитива будет true при наличии запрашиваемого критерия, и false при отсутствии критерия.

10.4.1.2 Пример использования Mnemonic Read

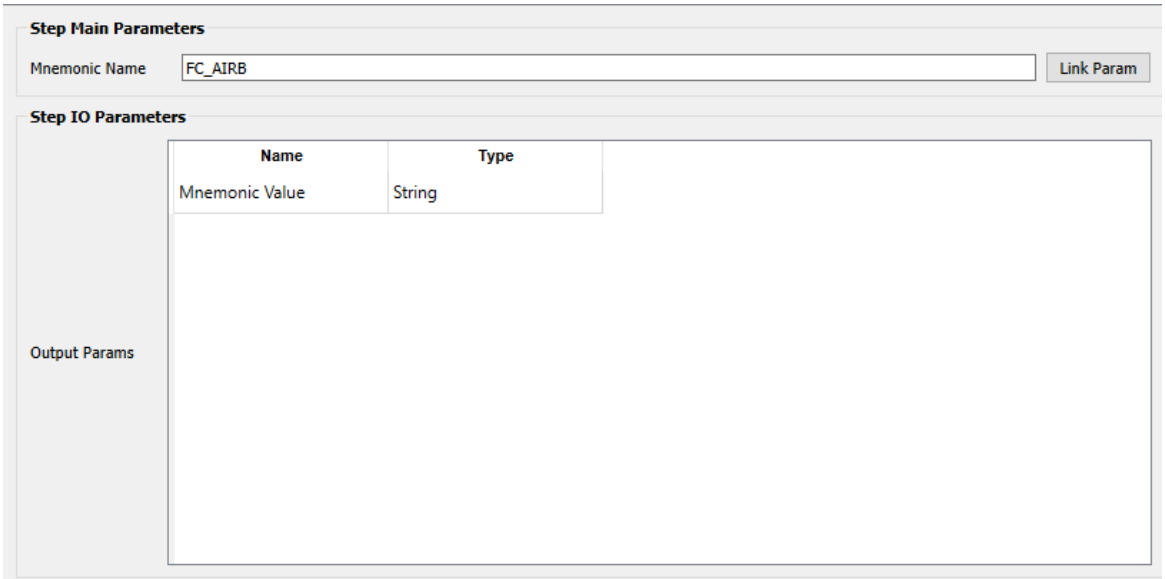
Для того, чтобы исполнительная программа (TSE) смогла считать значение запрашиваемой мнемоники или наличия критерия необходим доступ к соответствующему DEV файлу. Настройка доступа к DEV файлу осуществляется в TSE.

Пример использования:

1. В EOL Designer составим минимальную тестовую последовательность, состоящую из одного примитива Mnemonic Read:



2. В окне Specific Parameters напишем имя одной из мнемоник, которая есть в DEV файле:



3. Собираем проект нажав Build.
4. Запускаем TSE и видим трафик со значением запрашиваемой мнемоники т результатом примитива:

```
[INFO]    ------------------> Start "new_seq" Step "Read Mnemonic "FC_AIRB"" <-----------------

[INFO]    Block: "MnemonicRead.dll"

[INFO]    Mnemonic "FC_AIRB" = "8450043064"

[INFO]    Step result: TRUE

[INFO]    ------------------> Finish "new_seq" Step "Read Mnemonic "FC_AIRB"" <-----------------
```

10.4.2 Mnemonic Write

Данный примитив предназначен для записи значения мнемоники в DEV последовательность.

10.4.2.1 Specific parameters (Mnemonic Write)

В данном окне в поле *Mnemonic* необходимо написать (или прилинковать с помощью кнопки Link) имя мнемоники, служебную информацию и значение мнемоники.

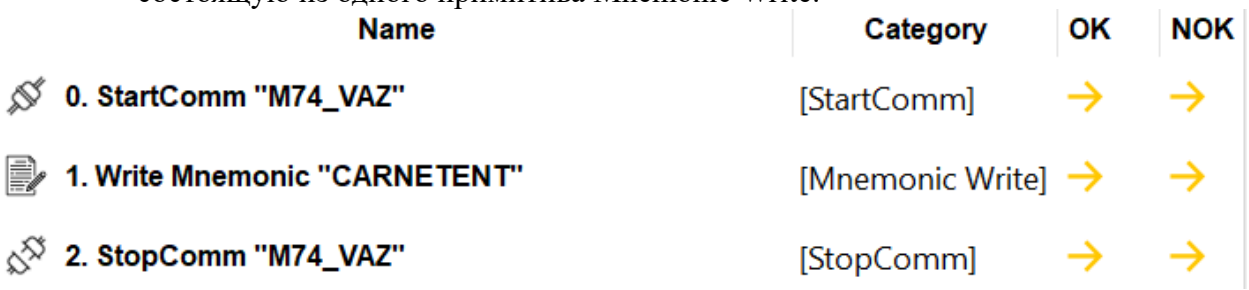
Если записанное имя мнемоники присутствует в DEV последовательности, то примитив обновит ее значение и результат примитива - true. Если мнемоника отсутствует, то результат примитива – false.

10.4.2.2 Пример использования Mnemonic Write

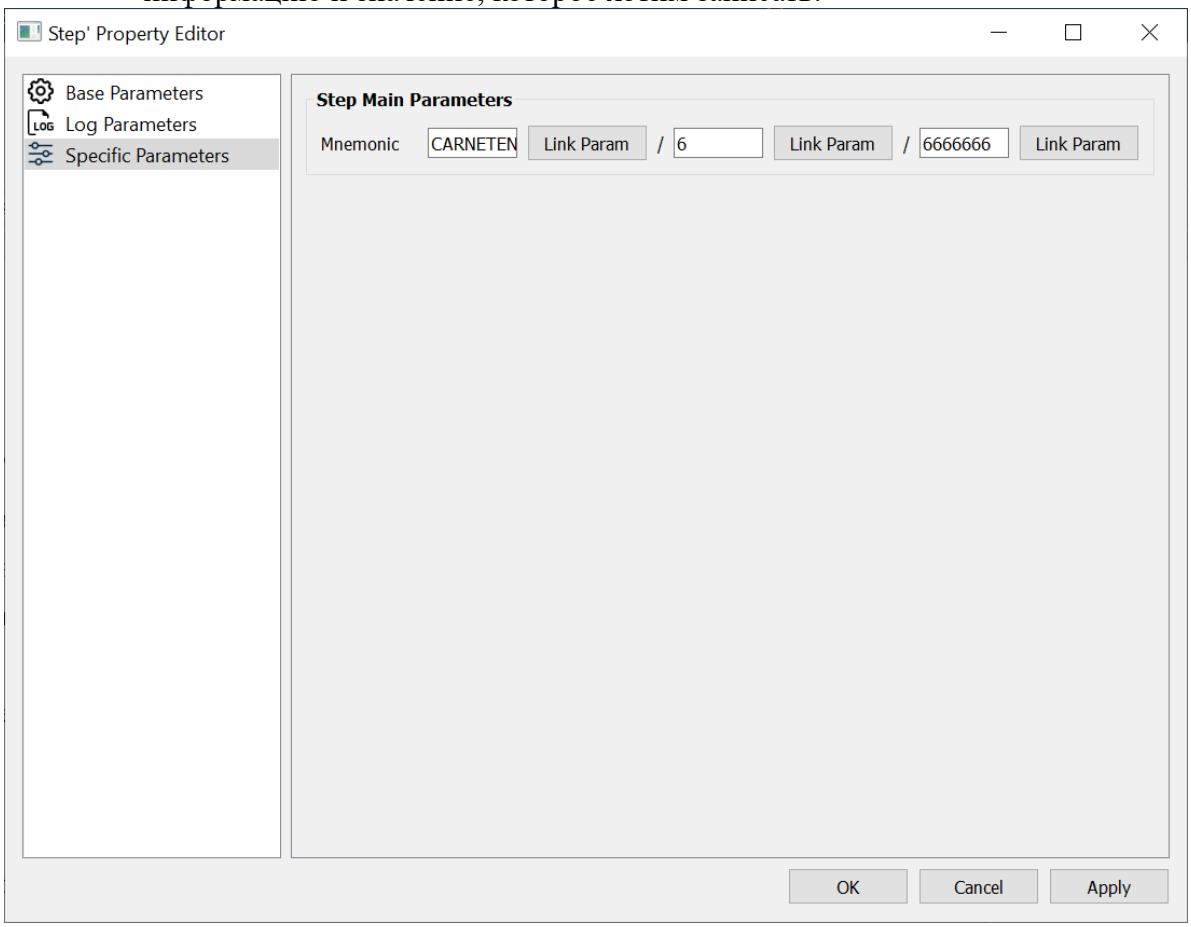
Для того, чтобы исполнительная программа (TSE) смогла считать значение запрашиваемой мнемоники или наличия критерия необходим доступ к соответствующему DEV файлу. Настройка доступа к DEV файлу осуществляется в TSE.

Пример использования:

1. В EOL Designer составим минимальную тестовую последовательность, состоящую из одного примитива Mnemonic Write:



2. В окне Specific Parameters напишем имя одной из мнемоник, служебную информацию и значение, которое хотим записать.



5. Собираем проект нажав Build.
6. Запускаем TSE и видим трафик:
7. Обновленный DEV файл выглядит следующим образом:

```
109225007310/XTAGFK210P0680002/CEVECP/XGF/KGF/E1/M4/ESS/RUS5/DG/GDFROI/TR4X2/DA/
AFURGE/SSRV/ABS/CA/PBCH/RETROE/SPROJA/RALU17/ACCAV/AIRBA1/SERIE/AVTOVA/SDPRPN/SA
BLAV/SSESP/FMMANU/EVMANU/SOPC1C/SPRGPL/SCHAD/LVAVEL/LVAREL/SGACHA/NOTRUS/00RUS/A
PL11/SRADIO/BVI5/CE11RU/ECLHB1/ENPBL1/TL21/SAN613/VSUVE/E02B/M2020/PHAS2/SAN913/
SSHYB/P3M/FDIU1/NCALL/MET03/NOBSD/ITPK1/VOLNCH/NOCAM/NUM_ENCH:6:1559/BOITE:6:149
9331/CALIBR_DAE:6:285044586R/CARNETENT:6:6666666/CR_PEVDYNA_BON:C:0/CR_PEVDYNA_FA
IT:C:0/CR_PEVSIGN_BON:C:0/CR_PEVSIGN_FAIT:C:0/CR_PEVSTAT1_BON:C:0/CR_PEVSTAT1_FA
IT:C:0/CR_PEVSTAT2_BON:C:0/CR_PEVSTAT2_FAIT:C:0/FC_AAP:6:8450042370/FC_ABS:6:845
0043345/FC_AIRB:6:8450043064/FC_DAE:6:8450030626/FC_TCU:6:8450042847/FC_TDB:6:84
50042210/MOTEUR:6:7090197/NOTUTIL:6:NOTRUS/PNBAT12V:6:8450033802/REFBOITEV:6:845
0120034/REFMOTEUR:6:8450120415/REF_AAP:6:8450040614/REF_ABS:6:8450043308/REF_AIR
B:6:8450040999/REF_DAE:6:8450040236/REF_INJECTION:6:8450034682/REF_TDB:6:8450043
329/REF_TRAIN_AR:6:8450032293/STRESS_AR:E:/STRESS_AV:E:/TRACA_TRAIN_AR:6:/UAG_ID
BAT:6:/UAG_IDCORA:6:/UAG_IDFSEL:6:/UAG_IDFSER:6:/UAG_IDFSLH:6:/UAG_IDFSRH:6:/UAG
_IDINJ:6:/ANTIDEM_T4:3:/DERNIER_IP:6:/ID_BADGE_CLE:3:/ID_UNITAIRE_INJ:6:/REFOP_A
AP:6:/REFOP_AIRB:6:/REFOP_HFM:6:/REFOP_INJECTION:6:/REFOP_USM:6:/RUC_AAP:6:/RUC_
ABS:6:/RUC_AIRB:6:/RUC_DAE:6:/RUC_HFM:6:/RUC_TCU:6:/RUC_TDB:6:/RUC_UCH:6:/T_R_AA
P:6:/T_R_ABS:6:/T_R_AIRB:6:/T_R_CLIM:6:/T_R_DAE:6:/T_R_HFM:6:/T_R_TDB:6:/T_R_UCH
:6:/T_R_USM:6:/UBCM:6:84500434.07F/
```

10.4.3 Get VIN

Данный примитив предназначен для считывания значения VIN из в DEV последовательности.

10.4.3.1 Specific parameters (Get VIN)

В данном окне ничего заполнять не требуется.

Если примитиву удалось считать VIN из DEV файла, то результат примитива true, если не удалось, то false.

10.5 Expressions

Данный раздел примитивов выполняет функции калькуляции с различными типами данных. В таблице 10.1 представлены характеристики доступных типов данных.

Тип данный	Пример	Доступные математические операции	Комментарии
Boolean	true, false	== != && ||	Данный тип может взаимодействовать только со своим типом. Результат взаимодействия true или false.
Integer	Числовые значения без дробных значения	== != + - * / ^ > <	Данный тип может взаимодействовать со своим типом и с типом double. Результат взаимодействия зависит от выбранного примитива. Если If Statement, то true или false (тип boolean). Если Expression, то integer или double в зависимости от выбора пользователя.

		>= <=	
Double	Числовые значения с дробными значениями	== != + - * / ^ > < >= <=	Данный тип может взаимодействовать со своим типом и с типом integer. Результат взаимодействия зависит от выбранного примитива. Если If Statement, то true или false (тип boolean). Если Expression, то integer или double в зависимости от выбора пользователя.
Array (integer)	Массив числовых значений без дробных значений	== !=	Данный тип может взаимодействовать только со своим типом.

Таблица 10.1 – характеристики типов данных

10.5.1 If Statement

Примитив If Statement выполняет функцию сравнения значений в соответствии с их типом данных. Результатом сравнения может быть состояние OK (true) или NOK (false). В зависимости от результата выбирается следующий шаг тестовой последовательности (см. настройки *Step Targets*).

Для сравнения используются следующие математические операции:

==	равно
!=	не равно
>	больше
>=	больше или равно
<	меньше
<=	меньше или равно

10.5.1.1 Base Parameters (If Statement)

Step Targets

Step Target принимает значение OK если результат сравнения был true, и NOK – если результат сравнения был false.

10.5.1.2 Specific Parameters (If Statement)

В данном окне формируется математическое выражение для сравнения, присваивается тип данных для каждого параметра и выбирается значение.

Поле *Expression* предназначено для математического выражения.

В поле *Input Params* выбираются типы параметров и их значения.

С помощью кнопки *Link Param* разработчик может в качестве данных использовать результат другого примитива.

Первоначально окно *Specific Parameters* примитива *If Statement* выглядит следующим образом:

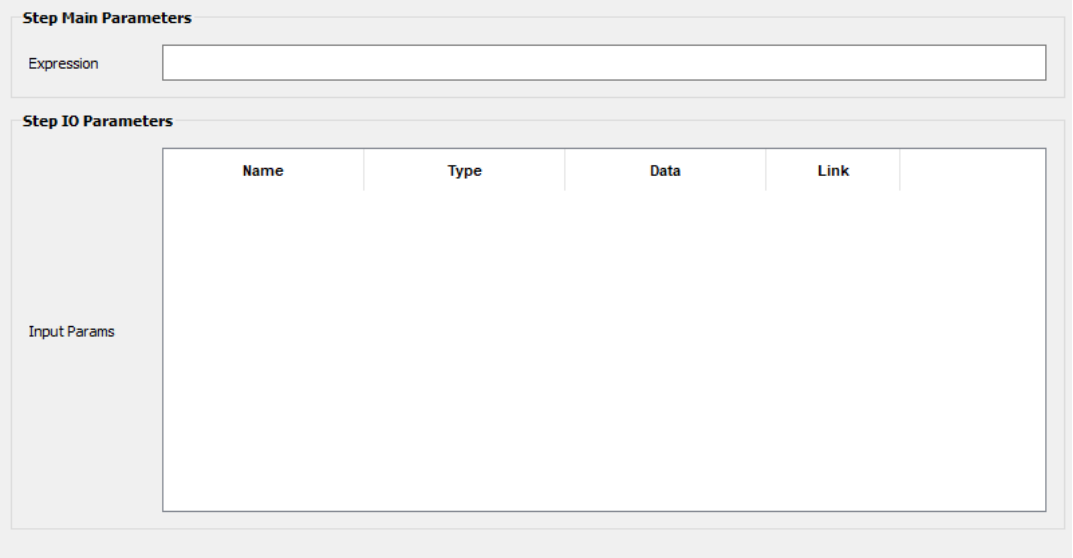
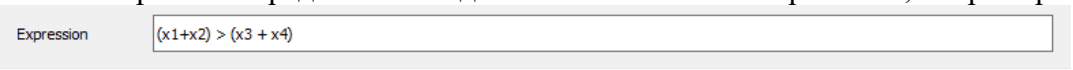
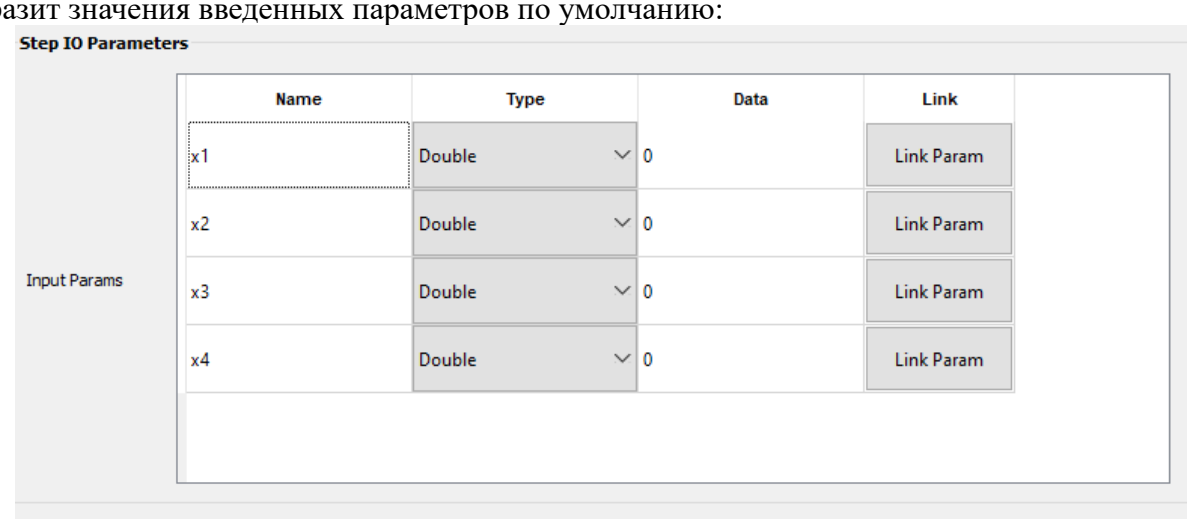


Рисунок 10.5 – Окно Specific Parameters примитива If Statement

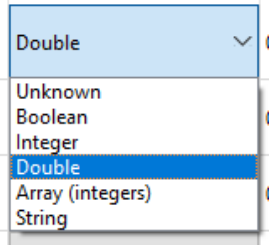
В поле Expression предназначено для математического выражения, например:



Затем, при нажатии клавиши *Enter* и в поле *Step IO Parameters* приложение отобразит значения введенных параметров по умолчанию:



Далее пользователь может изменить тип данных с помощью выпадающего списка для каждого параметра:



В столбце *Data* назначить конкретное значение или с помощью столбца *Link* присвоить выходное значение из любого другого примитива, выполненного ранее.

Если используется тип *Array (integers),* то список числовых значений задается через пробел.

10.5.1.3 Примеры использования If Statement

Пример сравнения типов Boolean представлен на рисунке 10.6. В данном примере *x1 = false*, *x2 = true*. В соответствии с полем *Expression* программа проверит математическое выражения *x1!=x2* на истинность. В данном случае *x1* действительно не равно *x2*, соответственно результатом примитива будет *true*.

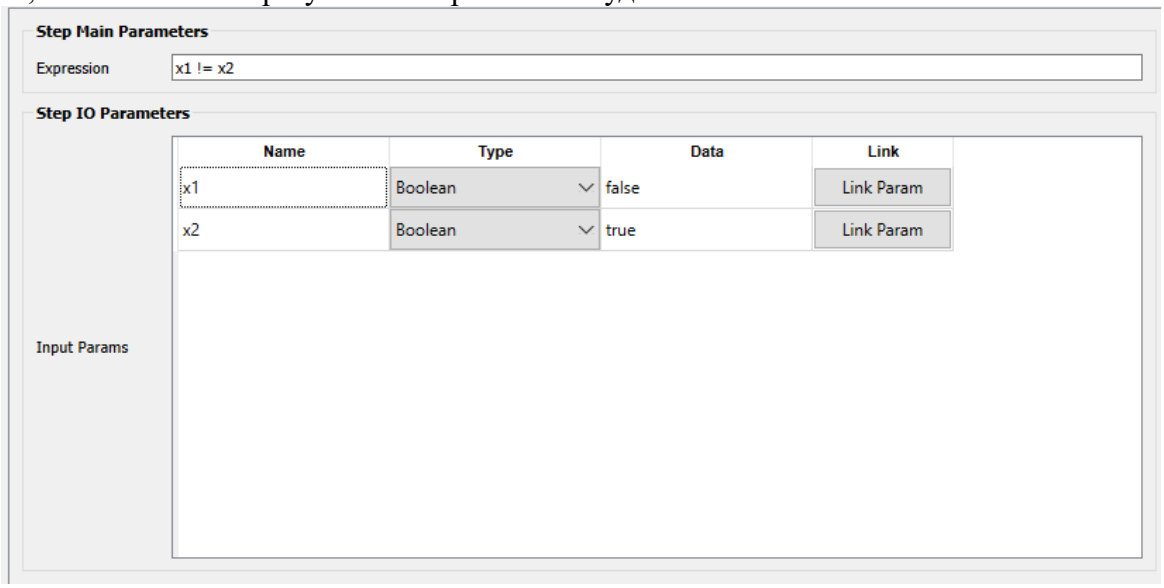


Рисунок 10.6 – Сравнение параметров типа boolean.

Пример сравнения типов integer представлен на рисунке 10.7. В данном примере *x1 = 3*, *x2 = 4*. В соответствии с полем *Expression* программа проверит математическое выражения *x1==x2* на истинность. В данном случае *x1* не равно *x2*, соответственно результатом примитива будет *false*.

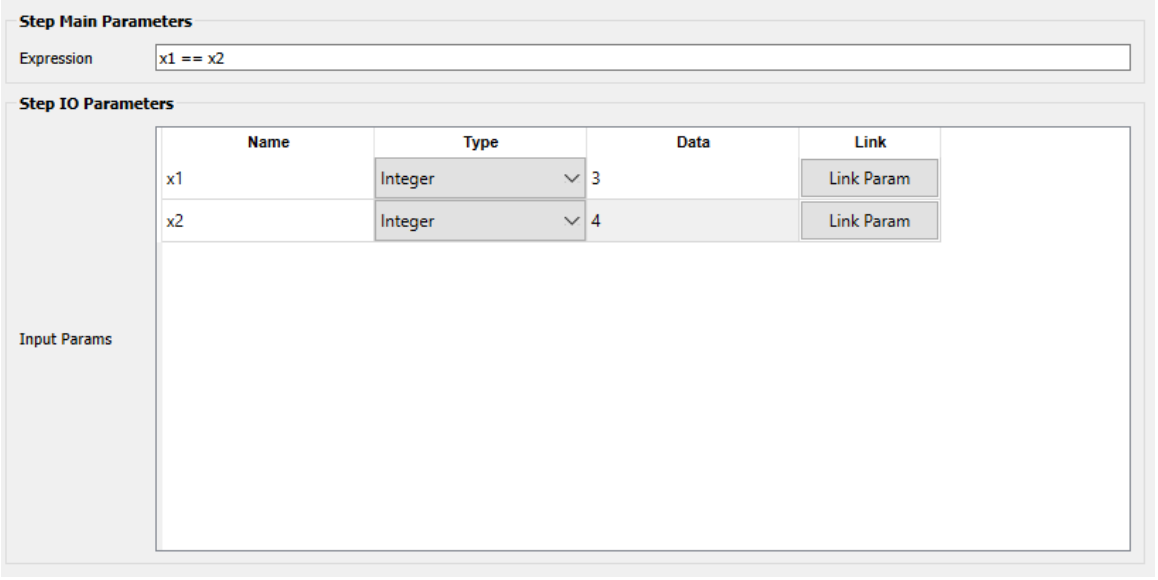


Рисунок 10.7 – Сравнение параметров типа integer.

Возможен комбинированный вариант сравнения, представленный на рисунке 10.8. В данному случае результатом сравнения *x1* и *x2* будет тип *boolean*. Соответственно далее обычное сравнение двух параметров типа *boolean*. Результатом работы данного примитива будет *true*.

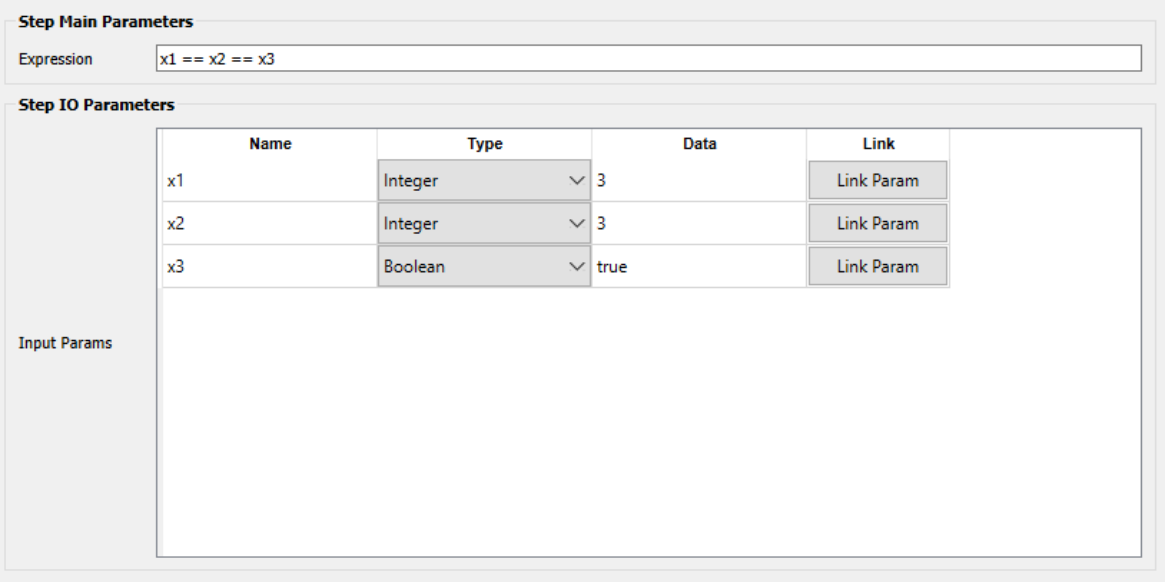


Рисунок 10.8 – Сравнение параметров типа integer и boolean.

Пример сравнения типа double представлен на рисунке 10.9. В данном случае *x1 = 11.5*, *x2 = 12.55*. В соответствии с полем *Expression* программа проверит математическое выражения *x1<x2* на истинность. В данном случае результатом примитива будет *true*.

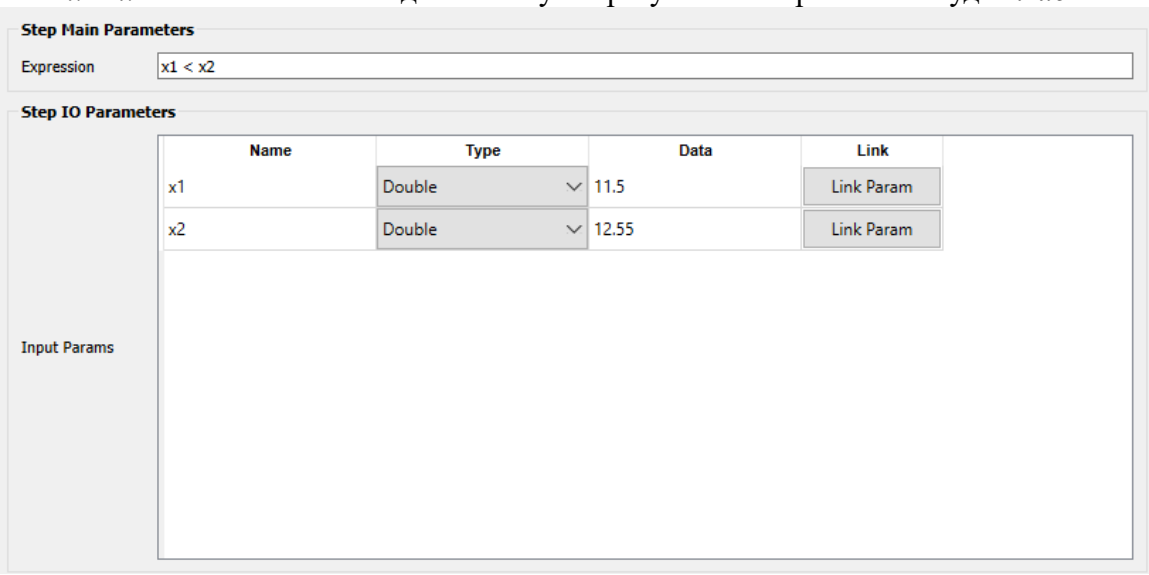


Рисунок 10.9 – Сравнение параметров типа double.

Также допускается сравнение параметра типа *integer* с параметром типа *double*.

Пример сравнения типа *array* представлен на рисунке 10.10. В данном случае массив *x1* не равен массиву *x2*. Соответственно результатом примитива будет *false*.

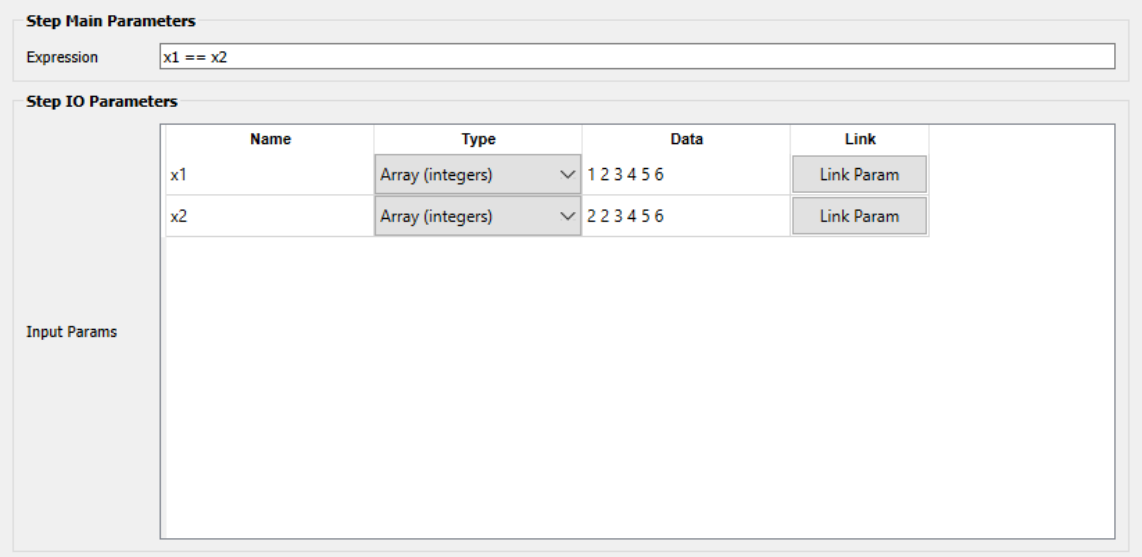


Рисунок 10.10 – Сравнение параметров типа array.

Есть возможность сравнить один элемент массива с числом или часть одного массива с другим. Например, если нужно сравнить нулевой элемент массива с числом, то в поле *Expression* выражение должно выглядеть следующим образом:
x1[0] == x2.

Если нужно выделить часть массива, запись формируется следующим образом: *x1[начальный элемент, конечный элемент]*. Пример сравнения части массива *x1* с массивом *x2* на рисунке 10.11.

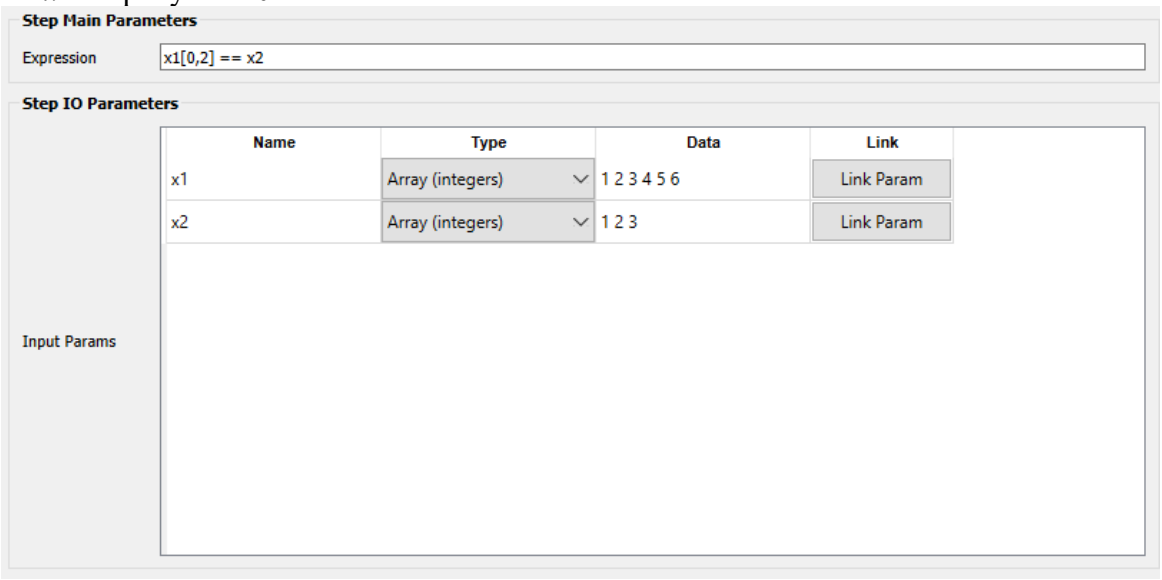


Рисунок 10.11 – Сравнение параметров типа array.

Пример сравнения типов integer представлен на рисунке 10.12. В данном примере *x1 = да*, *x2 = нет*. В соответствии с полем *Expression* программа проверит математическое выражения *x1==x2* на истинность. В данном случае результатом примитива будет *false*.

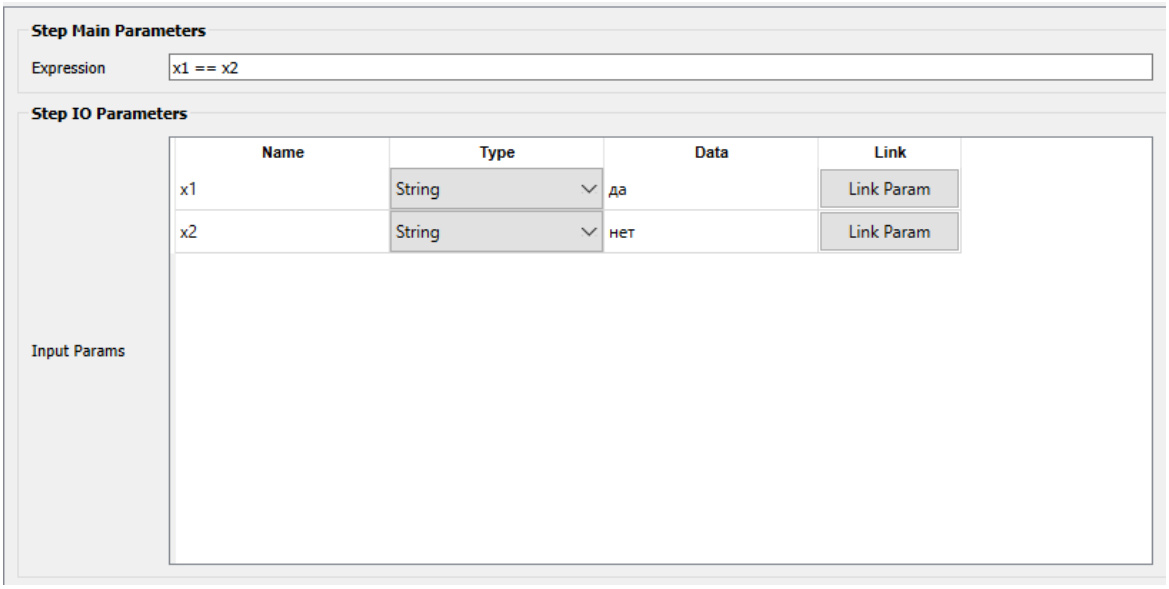


Рисунок 10.12 – Сравнение параметров типа string.

10.5.2 Expression

Примитив Expression предназначен для выполнения вычислительных функций, а также сравнения как If Statement.

Если Expression используется как калькулятор, то результатом примитива будет вычисленное значение.

Если Expression используется как If Statement, то результатом примитива будет true или false.

10.5.2.1 Specific Parameters (Expression)

В данном окне формируется математическое выражение для вычисления, присваивается тип данных для каждого параметра и выбирается значение. Первоначально окно *Specific Parameters* примитива *Expression* выглядит следующим образом:

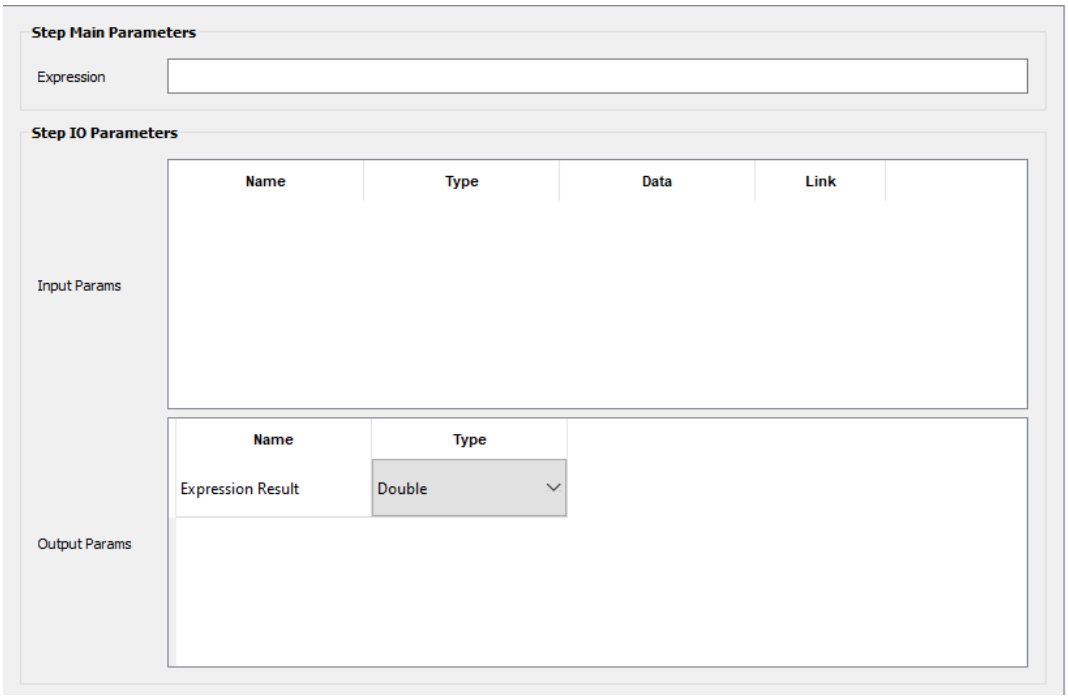
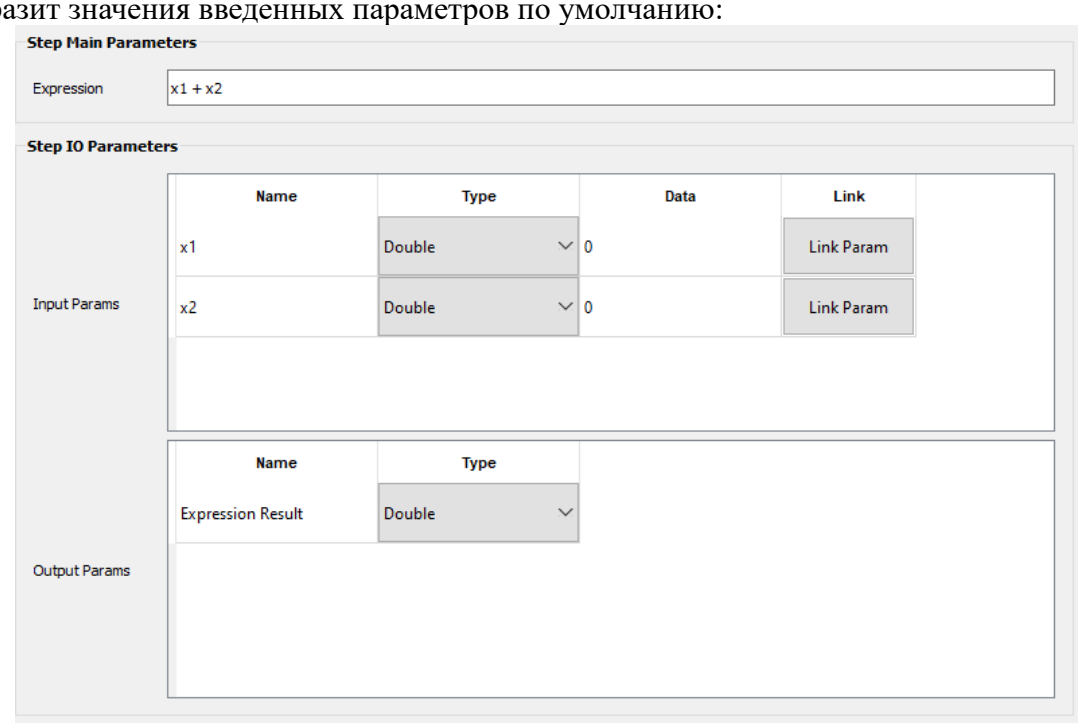


Рисунок 10.13 - Окно Specific Parameters примитива Expression.

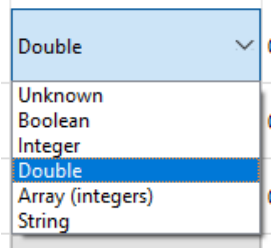
В поле Expression предназначено для математического выражения, например:



Затем, при нажатии клавиши *Enter* и в поле *Step IO Parameters* приложение отобразит значения введенных параметров по умолчанию:



Далее пользователь может изменить тип данных с помощью выпадающего списка для каждого параметра:



В столбце *Data* назначить конкретное значение или с помощью столбца *Link* присвоить выходное значение из любого другого примитива, выполненного ранее.

Если используется тип *Array (integers),* то список числовых значений задается через пробел.

10.5.2.2 Пример использования Expression

Для примера вычислим значение выражение ((x1+x2)*x3)^2, рисунок 10.14. Результатом вычисления будет 81.

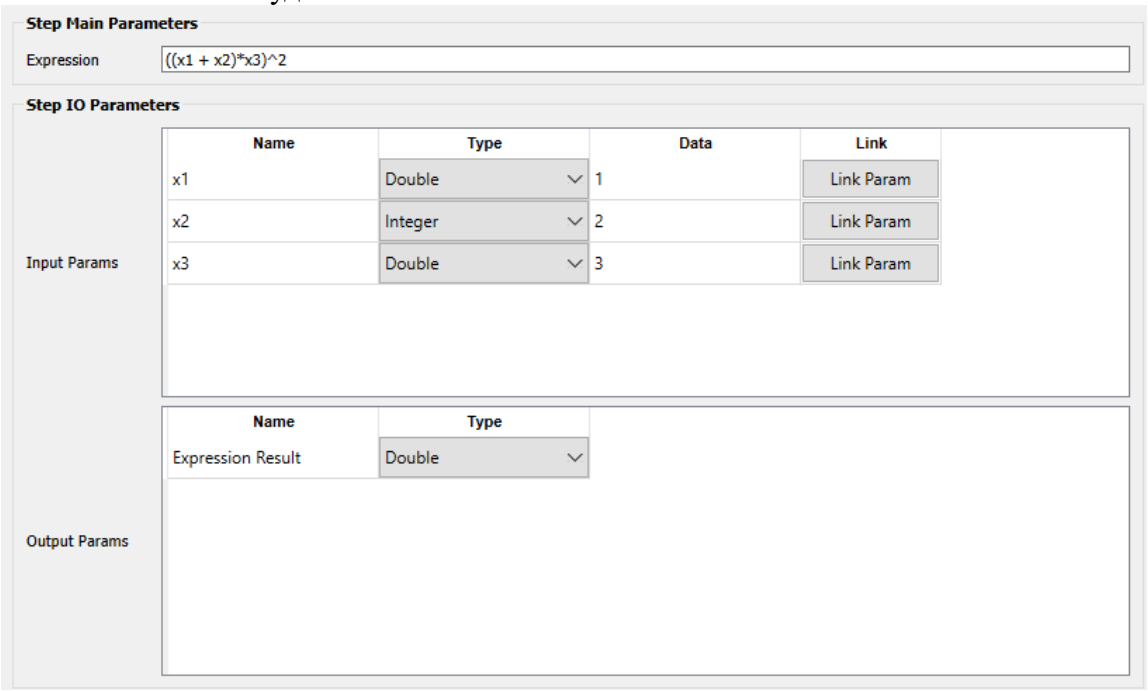


Рисунок 10.14 – Пример использования примитива Expression

10.6 Protocols

Данная категория примитивов осуществляет взаимодействие с различными ЭБУ посредством UDS запросов и ответов, основанных на стандарте *ISO 14229*.

10.6.1 ODX Message

В данном примитиве формируются диагностические сообщения на базе ODX файла ЭБУ.

10.6.1.1 Base Parameters (ODX Message)

Step Information имеет следующие поля:

Название поля	Описание
ODX File	Путь к ODX файлу, который необходимо загрузить.
ODX Params	Данное поле формируется из загруженного ODX файла и предлагает выбрать: DIAG-LAYER-CONTAINER; ЭБУ; Диагностический сервис.
Step Custom Name	Имя шага указанное разработчиком.
Step Id	Идентификатор шага. Генерируется автоматически.
Step Category	Заполняется автоматически.
Step Subcategory	Заполняется автоматически.
Step Description	Данное поле предназначено для комментариев разработчика.

Step Targets

Step Target принимает значение OK если TSE получил положительный ответ от ЭБУ в соответствии с диагностическим протоколом. Step Target принимает значение *NOK* если TSE не получил положительного ответа.

Название поля	Описание
Step If OK	Выбор следующего шага тестовой последовательности.
Step If NOK	Выбор следующего шага тестовой последовательности.

10.6.1.2 Input Parameters (ODX Message)

В данном окне отображаются параметры диагностического запроса в ЭБУ. Заполнение всех параметров автоматическое. Данные подгружаются из ODX файла выбранного ЭБУ.

Название поля	Описание
Param Number	Номера параметра в диагностическом сервисе. С помощью стрелочек  разработчик имеет возможность переходить от одного параметра к другому.
Param Name	Имя выбранного параметра.
Param Semantic	Атрибут параметра, обозначающий его смысл.
Param Type	Тип данных выбранного параметра.
Coded Value	Значение выбранного параметра.
DOP	Data Object Property (свойства объекта данных).
Unit	Тип кодирования DOP.
Compu Method	Метод расчета соответствия между внутренним типом и физическим типом диагностических значений.

Если сервис подразумевает отправление параметров в ЭБУ, то в поле *Linked Parameters* нужно заполнить раздел *Data*.

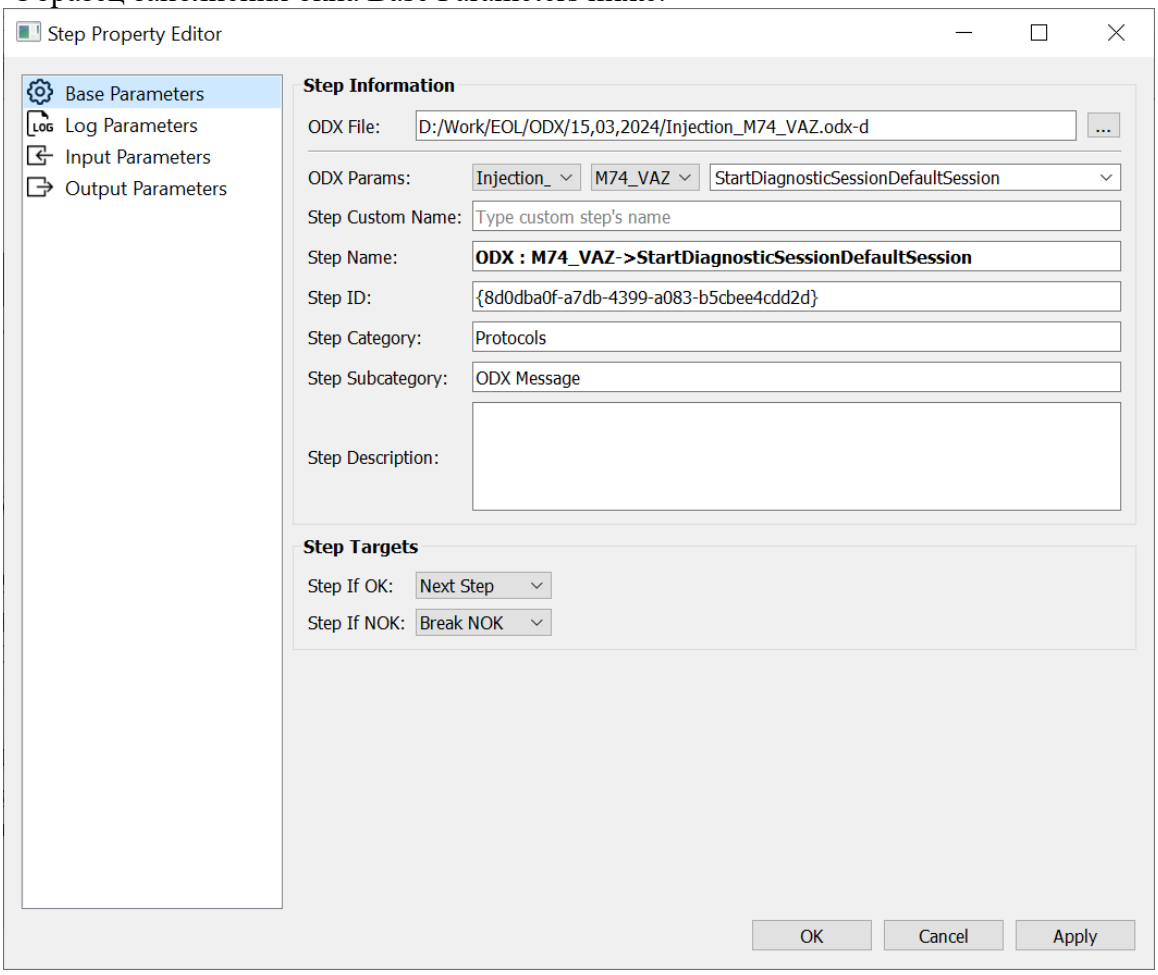
10.6.1.3 Output Parameters (ODX Message)

В данном окне отображаются параметры диагностического ответа от ЭБУ. Заполнение всех параметров автоматическое. Данные подгружаются из ODX файла выбранного ЭБУ. Все поля аналогичны Input Parameters.

10.6.1.4 Пример использования (ODX Message)

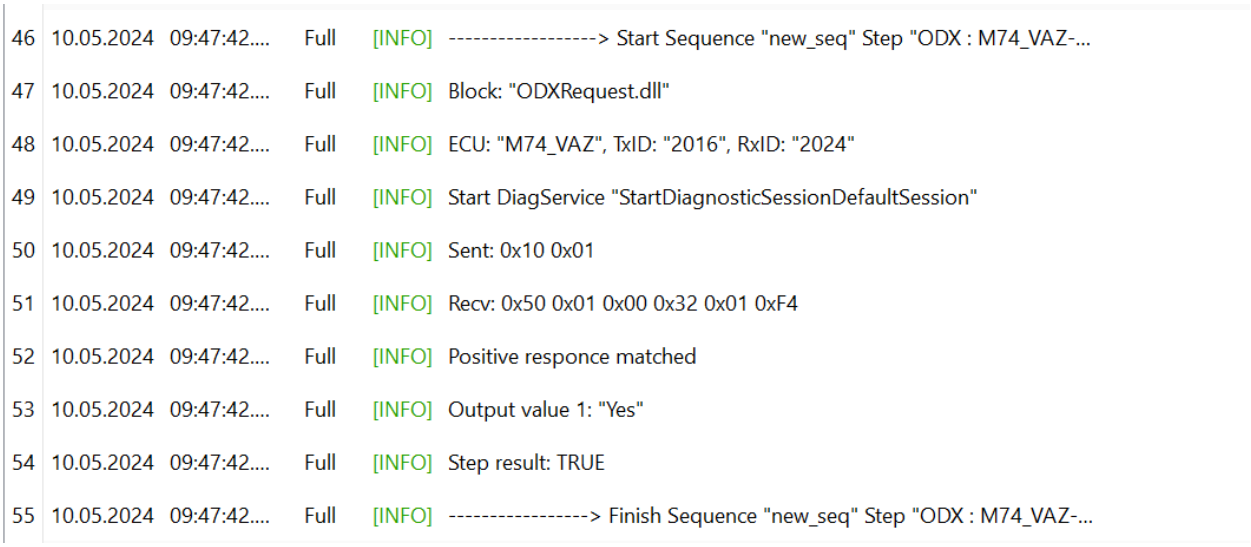
Для примера рассмотрим сервис StartDiagnosticSessionDefaultSession. В качестве ЭБУ используется M74.9.

Образец заполнения окна Base Parameters ниже:



В данном сервисе раздела Input Parameters и Output Parameters заполняются автоматически.

После сборки программы и запуска ее с помощью TSE видим следующий лог:



46	10.05.2024	09:47:42....	Full	[INFO]	----------> Start Sequence "new_seq" Step "ODX : M74_VAZ-...
47	10.05.2024	09:47:42....	Full	[INFO]	Block: "ODXRequest.dll"
48	10.05.2024	09:47:42....	Full	[INFO]	ECU: "M74_VAZ", TxID: "2016", RxID: "2024"
49	10.05.2024	09:47:42....	Full	[INFO]	Start DiagService "StartDiagnosticSessionDefaultSession"
50	10.05.2024	09:47:42....	Full	[INFO]	Sent: 0x10 0x01
51	10.05.2024	09:47:42....	Full	[INFO]	Recv: 0x50 0x01 0x00 0x32 0x01 0xF4
52	10.05.2024	09:47:42....	Full	[INFO]	Positive responce matched
53	10.05.2024	09:47:42....	Full	[INFO]	Output value 1: "Yes"
54	10.05.2024	09:47:42....	Full	[INFO]	Step result: TRUE
55	10.05.2024	09:47:42....	Full	[INFO]	----------> Finish Sequence "new_seq" Step "ODX : M74_VAZ-...

10.6.2 UDS Request

В данном примитиве формируются диагностические сообщения для ЭБУ в соответствии со стандартом *ISO 14229*.

10.6.2.1 Base Parameters (UDS Request)

Step Targets

Step Target принимает значение *OK* если был получен положительный ответ от ЭБУ, и *NOK* – если положительный ответ не был получен.

Название поля	Описание
Step If OK	Выбор следующего шага тестовой последовательности.
Step If NOK	Выбор следующего шага тестовой последовательности.

10.6.2.2 Specific Parameters (UDS Request)

Step Main Parameters

Название поля	Описание
CAN TX ID	Идентификатор сообщения, отправляемого тестовым оборудованием.
CAN RX ID	Идентификатор сообщения, принимаемого тестовым оборудованием.
CAN Baudrate	Скорость шины CAN.
Service Identifier	Идентификатор сервиса согласно UDS протоколу ISO 14229.
Subfunctions	Если сервис имеет подфункцию, то необходимо поставить галочку и ввести номер подфункции (опционально).

Step IO Parameters

В поле *Input Params* вводятся данные (*Data*) диагностического запроса, если есть такая необходимость, где:

- Name – наименование данных;
- Type – тип данных;

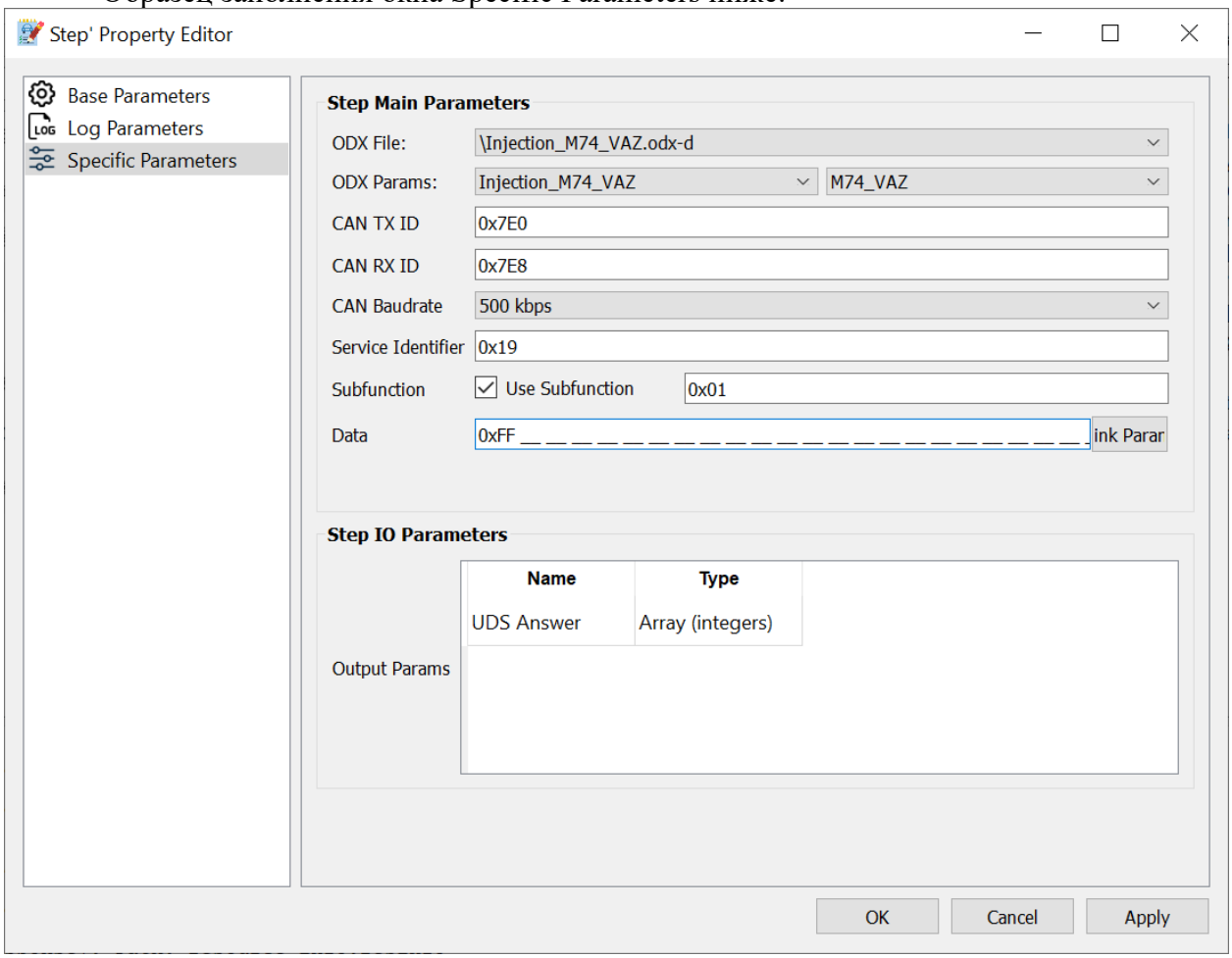
- Data – данные соответствующие диагностическому запросу. Данные должны вводиться в десятеричном формате через пробел.
- Link – с помощью данной кнопки разработчик имеет возможность в качестве данных выбрать результат любого другого предыдущего примитива.

В поле *Output Params* отображаются параметры диагностического ответа.

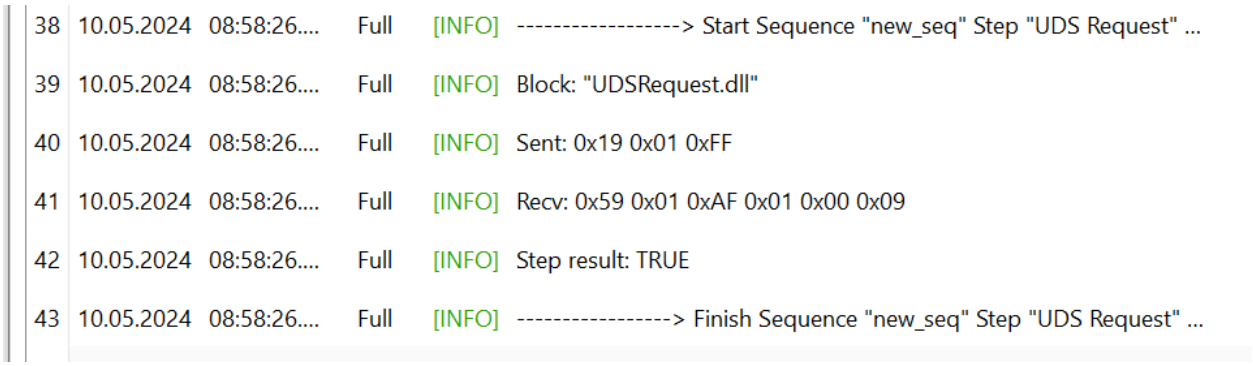
10.6.2.3 Пример использования (UDS Request)

Для примера рассмотрим сервис ReadDTCInformation. В качестве ЭБУ используется M74.9.

Образец заполнения окна Specific Parameters ниже:



После сборки программы и запуска ее с помощью TSE видим следующий лог:



10.6.3 Seed and Key

Данный примитив предназначен для реализации функции Security Access согласно ISO 14229.
Примитив Seed and Key реализует следующую очередность сервисов:

1. Diagnostic Session Control;
2. Request Seed;
3. Send Key.

10.6.3.1 Base Parameters (Seed and Key)

Step Targets

Step Target принимает значение OK если Security Access был пройден успешно, и NOK – если возникли ошибки при прохождении процедуры Security Access.

Название поля	Описание
Step If OK	Выбор следующего шага тестовой последовательности.
Step If NOK	Выбор следующего шага тестовой последовательности.

10.6.3.2 Specific Parameters (Seed and Key)

Для корректной работы примитива нужно правильно заполнить следующие поля:

Название поля	Описание
CAN TX ID	Идентификатор сообщения от системы EOL к ЭБУ.
CAN RX ID	Идентификатор сообщения от ЭБУ к системе EOL.
CAN Baudrate	Скорость шины.
Algorithm	Алгоритм расчета ключа.
Security Level	Уровень защищенности. Уровень защищенности зависит от диагностической сессии.
Diagnostic Session Control	Выбор диагностической сессии.
Request Seed subfunction	Сабфункция диагностического запроса Request Seed.
Request Seed random	

SA SK Number	Сабфункция диагностического запроса Send Key.
Request Seed SF	Random Value для запроса Reques Seed (всегда отправлять 0xAD).
Secure algorithm ddl path	Путь с библиотекой расчета ключа.

10.6.4 DTC Read

Диагностика электронных блоков управления автомобилем в первую очередь подразумевает чтение и анализ ошибок, т.н. диагностические коды неисправности (DTC – diagnostic trouble codes), для работы с которыми в протоколе UDS предусмотрен диагностический сервис ReadDTCInformation ($19).

В среде разработки тестовых последовательностей EOL Designer для работы с DTC предусмотрен примитив *DTC Read* (см. рисунок 10.15).

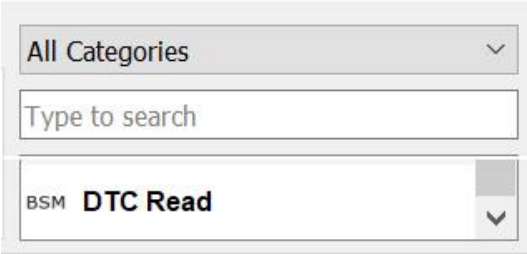
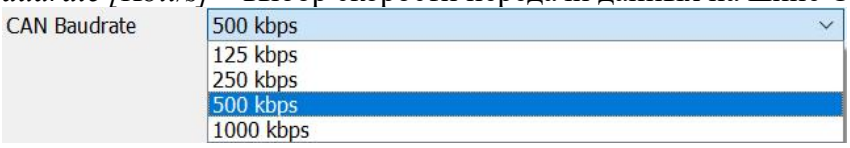


Рисунок 10.15 – Отображение примитива работы с диагностическими ошибками в области выбора примитивов

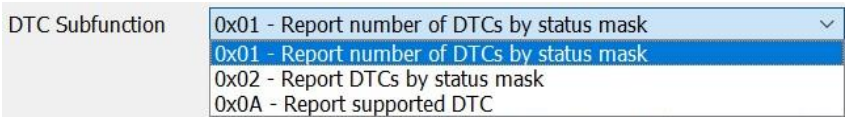
10.6.4.1 Описание параметров примитива

Ниже приведено описание параметров, настраиваемых в свойствах примитива во вкладке Specific Parameters (см. рисунок 10.16):

- *CAN TX ID* – диагностический id запроса к серверу (ECU);
- *CAN RX ID* – диагностический id сервера в ответ на запрос от клиента (MDA);
- *CAN Baudrate [Kbit/s]* – выбор скорости передачи данных на шине CAN;



- *DTC List File Name* – имя файла списка DTC “dtc_ECUname.xml”;
- *DTC Filter File Name* – имя файла фильтра DTC “dtcFilter_ECUname.xml”;
- *DTC Filter Name* – имя списка (группы) DTC из файла “dtcFilter_ECUname.xml”, которые подвергаются маскированию <IgnoreDTC> или проверке <CheckDTC>;
- *DTC Subfunction* – выбор подфункции сервиса $19:



- 0x01 – получить данные о количестве DTC по вычитываемой Status Mask;
- 0x02 – вычитать массив ошибок, соответствующих Status Mask;
- 0x0A – вычитать массив всех ошибок, которые содержатся в памяти ECU.

- *DTC Status Mask* – битовая маска статуса в запросах $19 $01 и $19 $02.

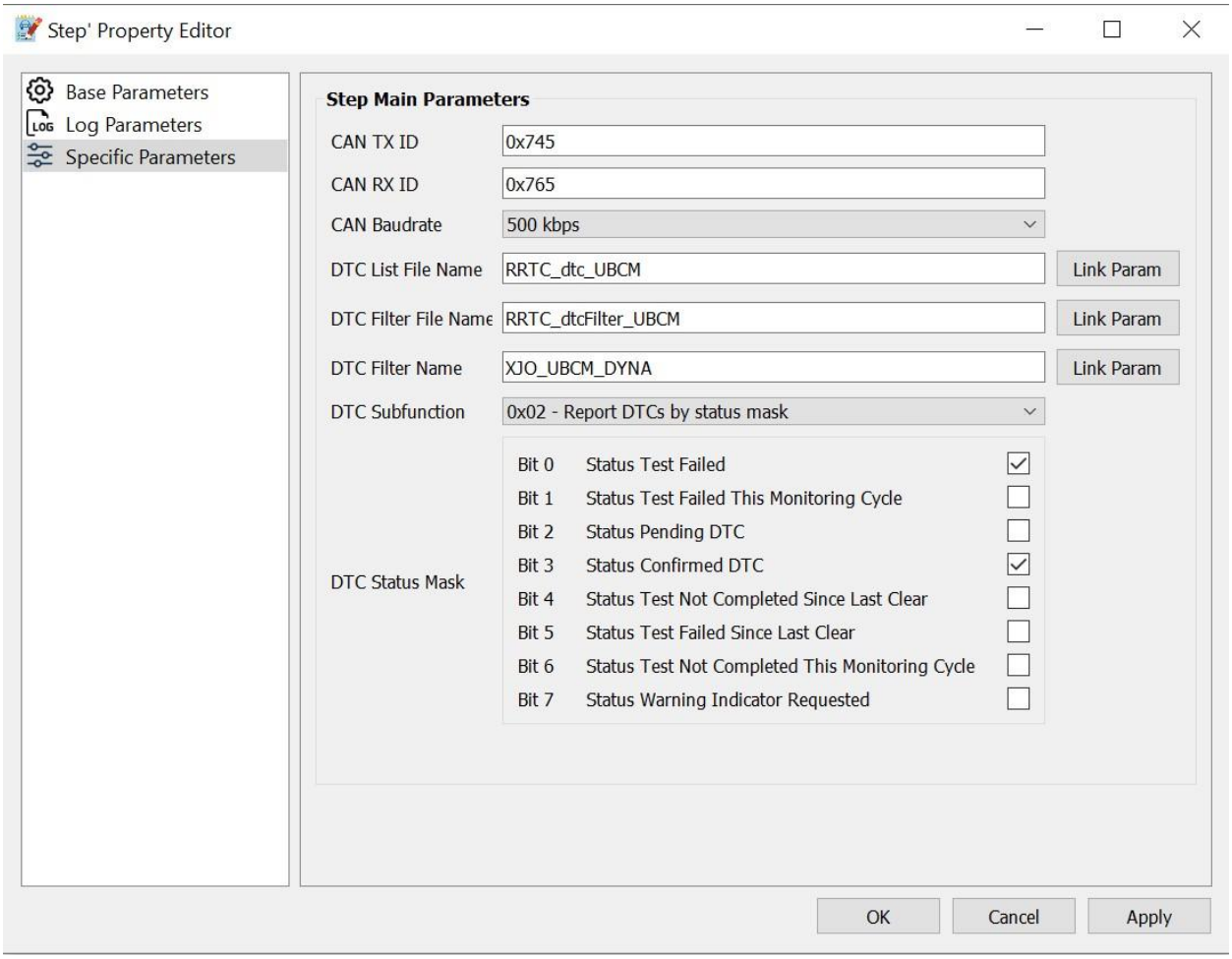


Рисунок 10.16 – Окно настройки параметров примитива *DTC Read*

10.6.4.2 Пример использования

На рисунок 10.17 приведена тестовая последовательность, состоящая из 3х степов примитива *DTC Read* в настройках которых (параметр *DTC Subfunction*) указаны подфункции 0x01 (степ 1), 0x0A (степ 2), 0x02 (степ 3).



Рисунок 10.17 – Шаги тестовой последовательности работы с примитивом *DTC read*

Результат степов 1 и 2 = TRUE, если клиент получил positive response от сервера. FALSE – если получен negative response или ответ вовсе не получен.

Результат степа 3 = FALSE, если получен хотя бы один DTC код. Если в ответе на запрос не содержится DTC, то результат степа = TRUE.

10.6.4.3 Просмотр логов

Проанализировать полученные ответы можно в WEB статистике (https://stat.avpnet.ru:2225/) на страничках «Просмотр CAN логов» (рисунок 10.18) и «Просмотр подробных логов» (рисунки 10.19, 10.20 ,10.22).

⇨ 21	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 01	DTC Read	15:39:01.490	00:00:26.517	Request		OK	19 01 09
⇦ 22	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 01	DTC Read	15:39:01.513	00:00:26.540	Response	00:00:00.023	OK	59 01 89 01 00 0F
⇨ 23	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 0A	DTC Read	15:39:01.524	00:00:26.551	Request		OK	19 0A
⇦ 24	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 0A	DTC Read	15:39:01.550	00:00:26.577	Response	00:00:00.026	OK	59 0A 00 00 00 50 AC 20 01 50 AC 20 23 50 AC 20 24 50 AC 22 01 50 AC 22 23 50 AC
⇨ 25	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 02	DTC Read	15:39:01.566	00:00:26.593	Request		OK	19 02 09
› ⇦ 26	0x765	0x745	DTC Read "RRTC_dtc_UBCM" 19 02	DTC Read	15:39:01.586	00:00:26.613	Response	00:00:00.020	OK	59 02 89 AF B0 14 09 AF A1 55 09 AF 00 11 09 AC 01 11 09 AC 28 11 09 AC B1 11 09

Рисунок 10.18 - Просмотр CAN логов степов 1, 2, 3

272	29.05.2024	15:39:01.490	Can	INFO	[0x745] (3) -> 0x19 0x01 0x09
273	29.05.2024	15:39:01.513	Can	INFO	[0x765] (6) <- 0x59 0x01 0x89 0x01 0x00 0x0F
274	29.05.2024	15:39:01.513	Full	INFO	Status availability mask: "0x89"
275	29.05.2024	15:39:01.513	Full	INFO	DTCs number which match the status mask: "15"
276	29.05.2024	15:39:01.514	Full	INFO	Step result: TRUE

Рисунок 10.19 - Просмотр подробного лога степа 1

285	29.05.2024	15:39:01.524	Can	INFO	[0x745] (2) -> 0x19 0x0A
^ 286	29.05.2024	15:39:01.550	Can	INFO	[0x765] (150) <- 0x59 0x0A 0x00 0x00 0x00 0x50 0xAC 0x20 0x01 0x50 0xAC 0x20 0x23 0x50 0xAC 0x20 0x24 0x50 0xAC 0x22 0x01 0x50 0xAC 0x22 0x23 0x50 0xAC 0x22 0x24 0x50 0xAC 0x27 0x23 0x50 0xAC 0x24 0x23 0x50 0xAC 0x24 0x24 0x50 0xAC 0xD3 0x23 0x50 0xAC 0xD1 0x23 0x50 0xAC 0x99 0x09 0x50 0xAC 0x9B 0x09 0x50 0xAF 0xA6 0x94 0x00 0xAF 0xA7 0x94 0x00 0xAF 0xA0 0x4B 0x50 0xAC 0x25 0x23 0x50 0xAC 0x25 0x24 0x50 0xAF 0xA5 0x94 0x00 0xAF 0x10 0x23 0x50 0xAF 0x12 0x23 0x50 0xAF 0x03 0x04 0x50 0xAC 0xB4 0x64 0x50 0xAF 0xB8 0x01 0x50 0xAF 0xA8 0x01 0x50 0xAF 0xA1 0x01 0x50 0xAC 0x27 0x52 0x50 0xAC 0xD6 0x04 0x50 0xAC 0xB2 0x64 0x50 0xAC 0xD5 0x23 0x50 0xAF 0xA1 0x51 0x00 0xAF 0xA1 0x55 0x2F 0xAF 0x04 0x09 0x50 0xAF 0x05 0x09 0x50 0xAC 0x99 0x11 0x50 0xAC 0x9B 0x11 0x50
287	29.05.2024	15:39:01.550	Full	INFO	DTC "0x00" - "No Device found in DTC list : УСТРОЙСТВО И ОШИБКИ ODB";
288	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2001" - "AC20 ДАТЧИК ПЕДАЛИ ТОРМОЗА I_BS_SW : ОБЩАЯ ЭЛЕКТРИЧЕСКАЯ ОШИБКА";
289	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2023" - "AC20 ДАТЧИК ПЕДАЛИ ТОРМОЗА I_BS_SW : СЛАБЫЙ СИГНАЛ";
290	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2024" - "AC20 ДАТЧИК ПЕДАЛИ ТОРМОЗА I_BS_SW : СИЛЬНЫЙ СИГНАЛ";
291	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2201" - "AC22 ДАТЧИК ПЕДАЛИ ТОРМОЗ I_BLS_SW : ОБЩАЯ ЭЛЕКТРИЧЕСКАЯ ОШИБКА";
292	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2223" - "AC22 ДАТЧИК ПЕДАЛИ ТОРМОЗ I_BLS_SW : СЛАБЫЙ СИГНАЛ";
293	29.05.2024	15:39:01.550	Full	INFO	DTC "0xAC2224" - "AC22 ДАТЧИК ПЕДАЛИ ТОРМОЗ I_BLS_SW : СИЛЬНЫЙ СИГНАЛ";

Рисунок 10.20 - Просмотр подробного лога степа 2



334	29.05.2024	15:39:01.566	Can	INFO	[0x745] (3) -> 0x19 0x02 0x09
^ 335	29.05.2024	15:39:01.586	Can	INFO	[0x765] (63) <- 0x59 0x02 0x89 0xAF 0xB0 0x14 0x09 0xAF 0xA1 0x55 0x09 0xAF 0x00 0x11 0x09 0xAC 0x01 0x11 0x09 0xAC 0x28 0x11 0x09 0xAC 0xB1 0x11 0x09 0xAF 0x11 0x11 0x09 0xAC 0x05 0x13 0x09 0xAC 0x94 0x13 0x09 0xAC 0x95 0x13 0x09 0xAC 0x70 0x13 0x09 0xAC 0x9C 0x13 0x09 0xAC 0x9D 0x13 0x09 0xAC 0xA0 0x13 0x09 0xAC 0x90 0x13 0x89
336	29.05.2024	15:39:01.586	Full	INFO	Status avaibility mask: "0x89"
337	29.05.2024	15:39:01.586	Full	WARNING	DTC "0xAFB014" - "AFB0 ВХОД ПИТАНИЯ 02 : ОБРЫВ ЦЕПИ ИЛИ К.З. НА ЗЕМЛЮ";
338	29.05.2024	15:39:01.586	Full	WARNING	DTC "0xAFA155" - "AFA1 BMS : НЕ КОНФИГУРИРОВАНО";
339	29.05.2024	15:39:01.586	Full	WARNING	DTC "0xAF0011" - "AF00 ВЫХОДНОЕ ПИТАНИЕ ESCL : К.З. НА ЗЕМЛЮ";
340	29.05.2024	15:39:01.586	Full	WARNING	DTC "0xAC0111" - "AC01 ВЫХОДНОЕ ПИТАНИЕ BTP : К.З. НА ЗЕМЛЮ";

Рисунок 10.21 - Просмотр подробного лога степа 3

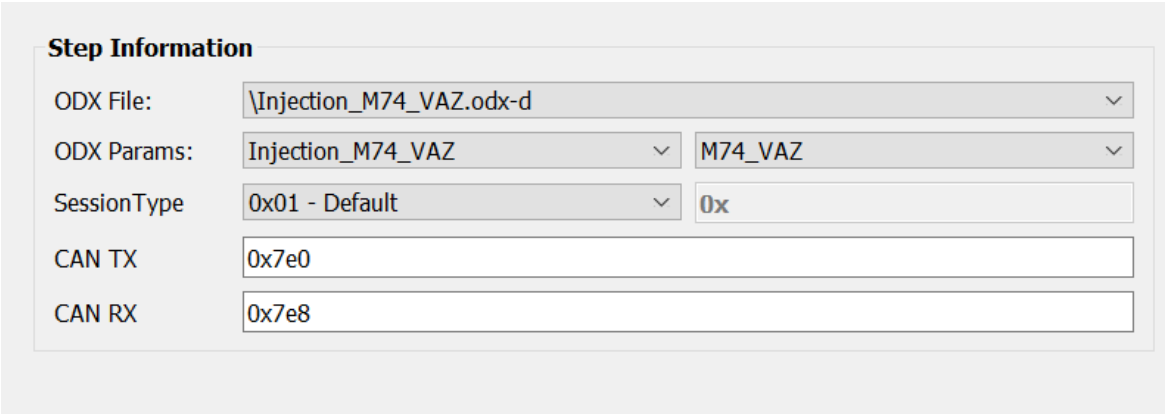
10.6.5 StartComm

Примитив предназначен для открытия сеанса связи с конкретным ЭБУ. Данный примитив является **обязательным** для начала сессии обмена между системой EOL и конкретным ЭБУ автомобиля.

Под сессией подразумевается период обмена диагностическими сообщениями с ЭБУ. Сессия обмена должна начинаться примитивом *StartComm* и заканчиваться *StopComm*. Недопустимо открытие одновременно двух сессий с одинаковыми ID. Одновременно может быть открыто до 16 сессий связи с различными ЭБУ.

10.6.5.1 Specific Parameters

В данном примитиве необходимо выбрать ODX файл соответствующего ЭБУ. Выбрать тип диагностической сессии

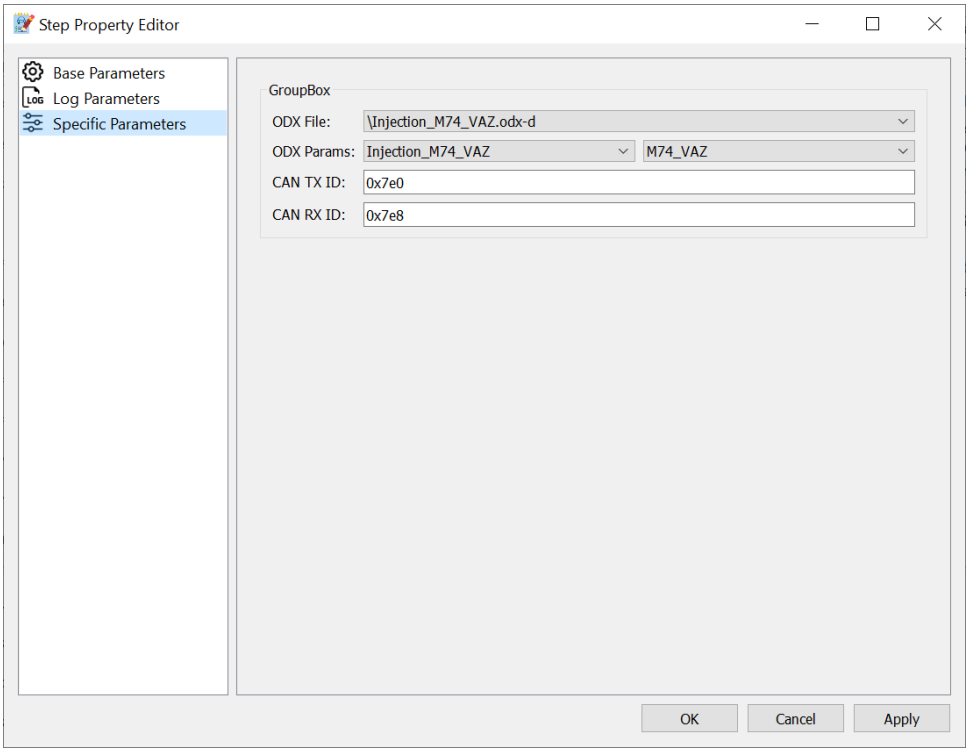


10.6.6 StopComm

Примитив предназначен для закрытия связи с ЭБУ. Данный примитив является обязательным для закрытия сессии обмена между системой EOL и конкретным ЭБУ автомобиля.

10.6.6.1 Specific Parameters

В данном примитиве необходимо выбрать ODX файл соответствующего ЭБУ.



10.7 Terminal

10.7.1 Operator Show Read

Примитив позволяет выводить на экран терминала сообщения для оператора.

10.7.1.1 Specific parameters (Operator Show Read)

В данном окне формируется сообщение оператору и возможные варианты ответа.

Поле *Message* предназначено для сообщения оператору. Сообщение может иметь как информационный характер, так и вопросительный. С помощью кнопки *Link Param* разработчик может в качестве сообщения использовать результат другого примитива. Если сообщение подразумевает ответ от оператора, то нужно добавить использование дополнительных кнопок с помощью *Add button*. Максимальное количество кнопок для добавления 8.

Поля *Button1*, *Button2*… *Button8* на экране терминала будут отображены как кнопки. Название кнопки соответствует наполнению соответствующего поля.

Если тестовая последовательность предназначена для запуска на планшете, то необходимо учитывать соответствие механических кнопок на планшете экранным:

Кнопка на планшете	Кнопка на экране
P1	Button 7
P2	Button 5
P3	Button 3
P4	Button 1
P5	Button 8
P6	Button 6
P7	Button 4
P8	Button 2

Frame color позволяет выбрать цвет фона для сообщения.

Для того, чтобы тестовая последовательность остановилась в ожидании ответа от оператора нужно поставить галочку *blocking version.*

10.7.1.2 Пример использования Operator Show Read

При заполнении полей окна Specific Parameters как показано на рисунке 10.22 на экране терминала должно появиться отображение как на рисунке 10.23.

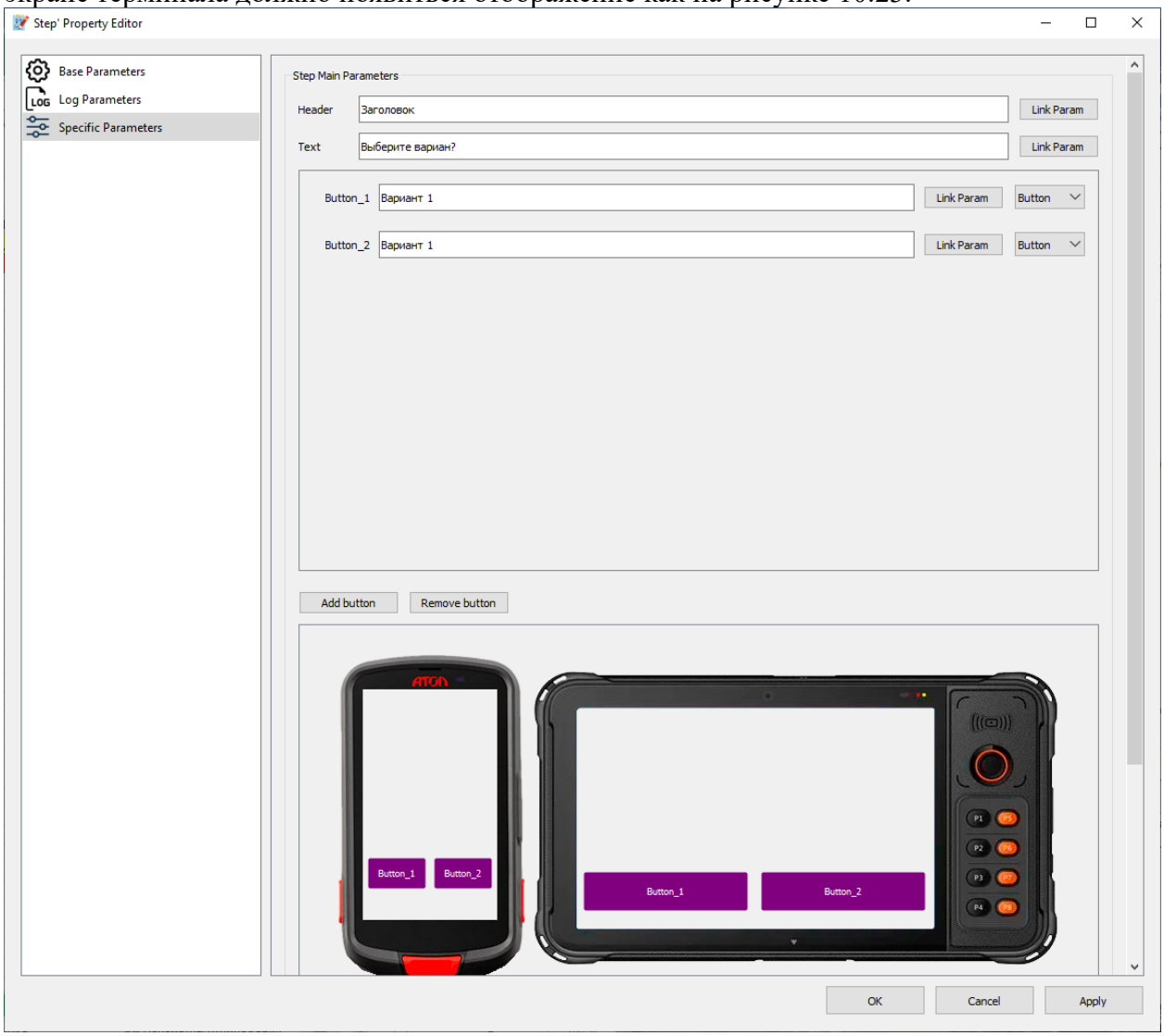


Рисунок 10.22 – Пример заполнения окна Specific Parameters примитива Operator Show Read.

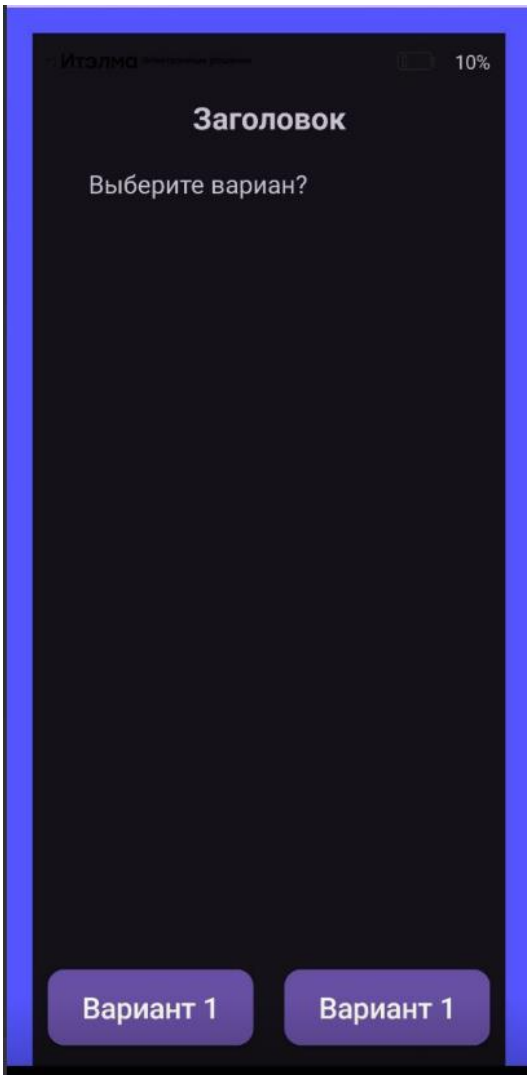


Рисунок 10.23 – Фактическое отображение на экране терминала при заполнении примитива как показано на рисунке 10.22