ООО «Новые Системы Электроники»

# АСУ Конфигуратор

## РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# Оглавление

# О программе

Одним из основных преимуществ использования программируемых силовых модулей компании «Новые Системы Электроники» является простота их программирования и перепрошивки. Это стало возможным благодаря среде АСУ Конфигуратор.

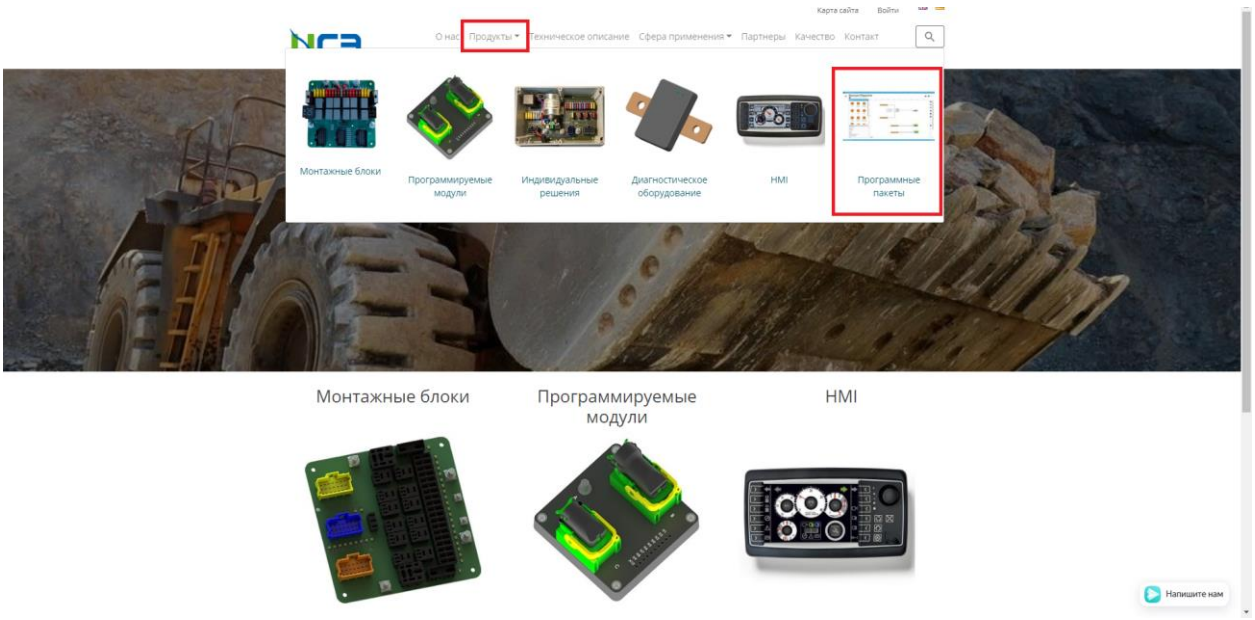
АСУ Конфигуратор – это система графического программирования, которая позволяет задавать логику работы силовых модулей, не написав ни одной строчки кода. Благодаря данной системе для создания алгоритма работы силовых модулей нет необходимости в изучении специальных языков программирования, достаточно лишь понимания основных логических операторов и математических функций. Программирование с помощью АСУ Конфигуратора напоминает рисование функциональной схемы.

Преимущества:

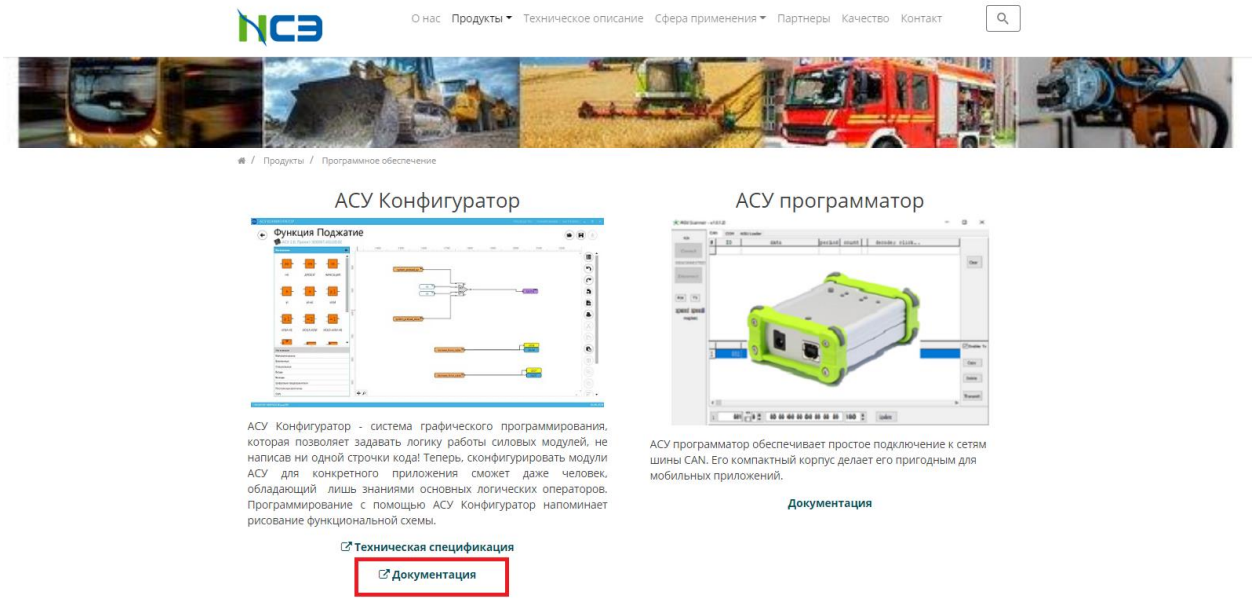
- Не требует специальных знаний
- Программирование происходит при помощи визуального интерфейса, напоминающего создание функциональной схемы
- Возможность быстрой корректировки алгоритма работы
- Максимальная простота использования шины CAN
- В зависимости от величины, программа может быть разбита на отдельные функциональные блоки
- Программа загружается в контроллер одним щелчком мышки
- Перепрограммирование блока может осуществляться без демонтажа
- Возможна одновременная работа с несколькими модулями

# Установка и запуск

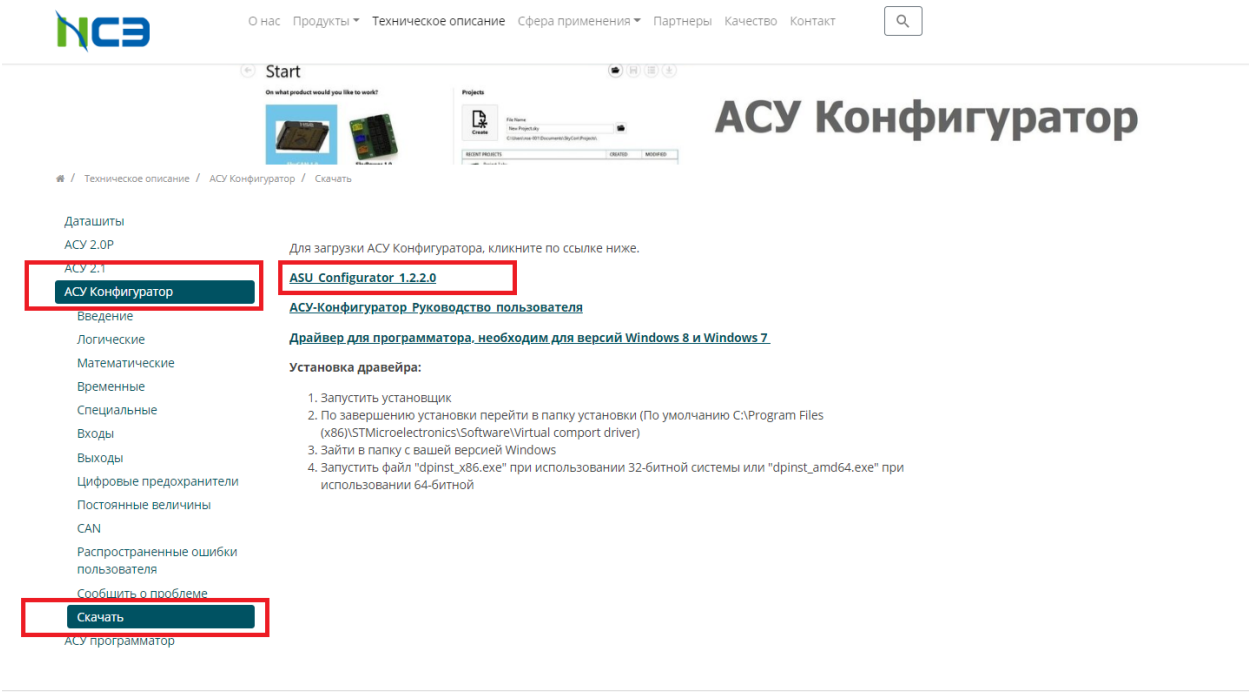
Для скачивания установщика необходимо перейти на сайт компании [www.nse-online.com](http://www.nse-online.com) и в разделе «Продукты» выбрать пункт «Программные пакеты» .



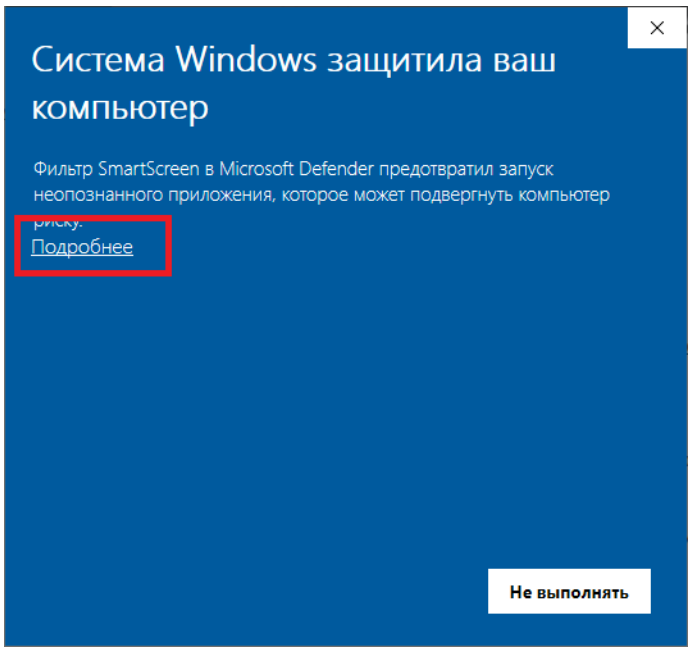

В открывшемся окне в разделе «АСУ Конфигуратор» выберите графу «Документация»



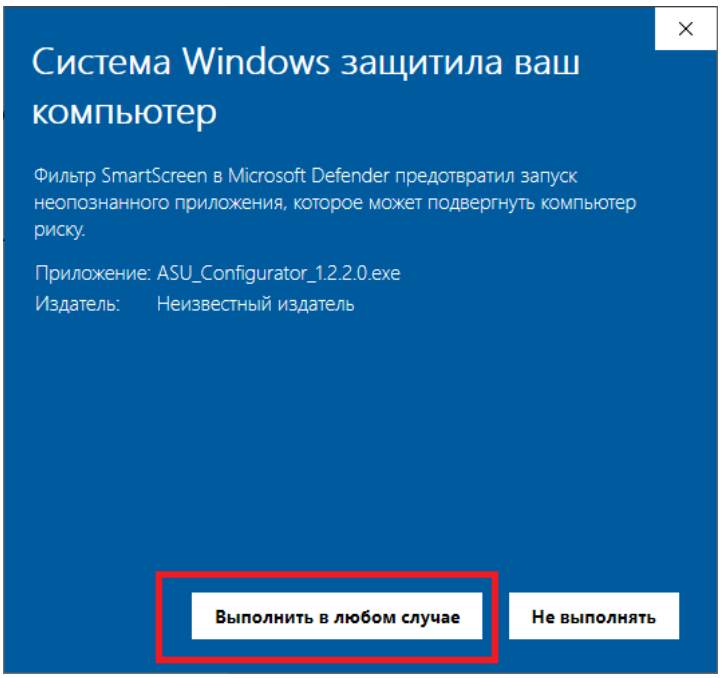
Далее перейдите в раздел «Скачать» и кликните по полю с названием программы для начала загрузки пакета.




После скачивания пакета, запустите установщик. Windows может проинформировать вас, что данный пакет получен из ненадежного источника.



Для продолжения установки кликните по ссылке «Подробнее» и нажмите «Выполнить в любом случае».




Установка программы пройдет следующие этапы:

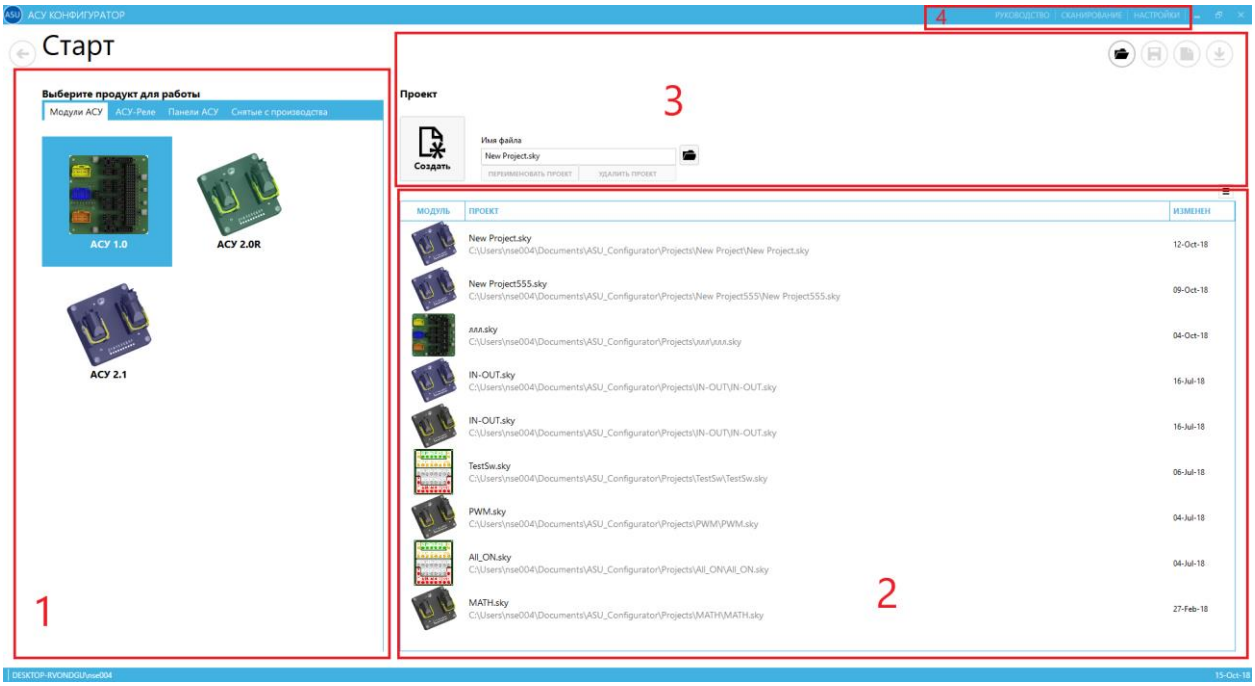
1. Установка основных компонентов АСУ Конфигуратора
2. Установка компилятора SDCC (необходим для работы без подключения к сети Интернет)
3. Установка пакета make-3.81 (необходим для работы без подключения к сети Интернет)
4. Проверка наличия и установка необходимых библиотек C++
5. Завершение установки, предложение о перезапуске компьютера для дальнейшей работы

# Открытие программы и настройки

## Главное окно программы

Главное окно программы условно можно разделить на 4 области:

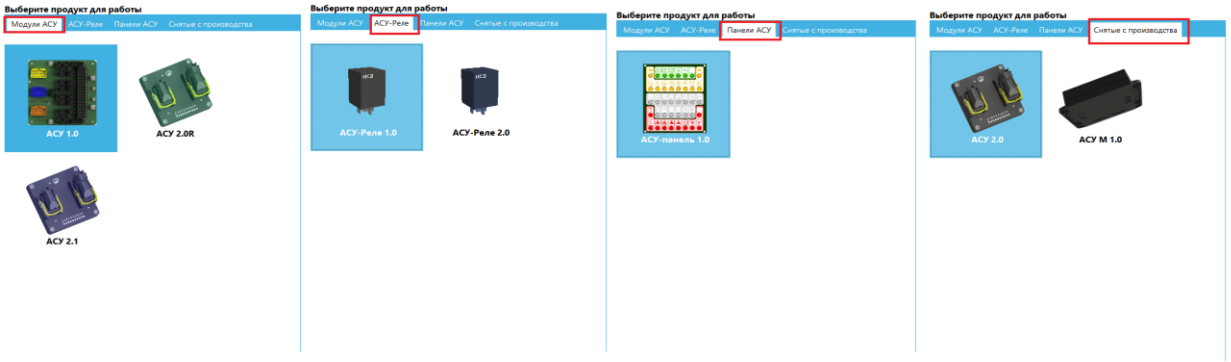
1. Модули и продукты компании
2. Ранее созданные проекты
3. Инструменты для создания и работы с проектами
4. Меню настроек, сканирование и руководство




Рассмотрим данные области более подробно.

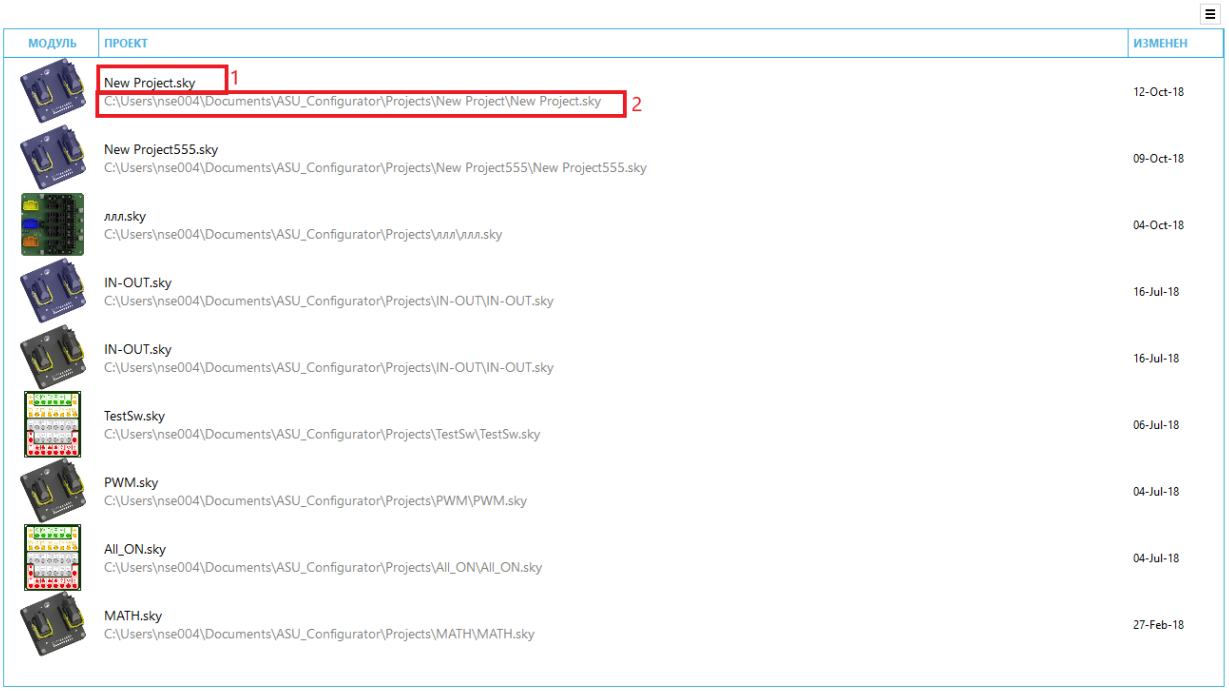
## Модули и продукты компании

В данном разделе пользователю доступен выбор модуля для создания проекта. Обратите внимание, что модули разделены на отдельные категории.

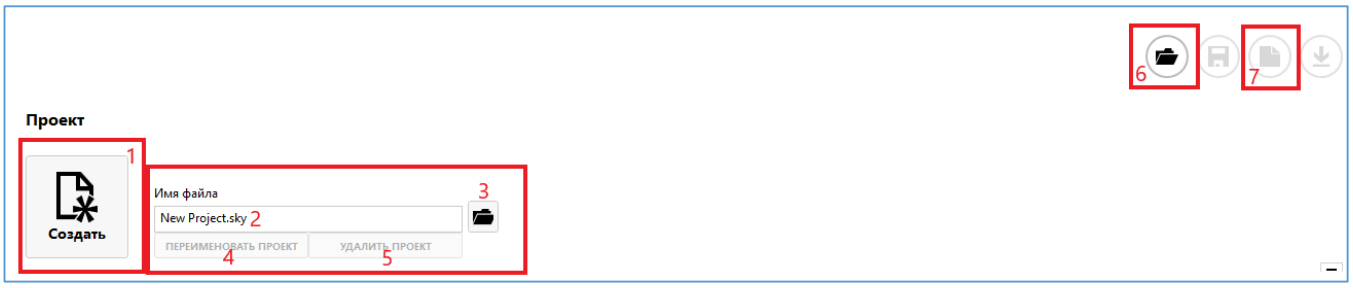



## Ранее созданные проекты

Здесь содержится информация о ранее созданных проектах, представленных в виде списка. По умолчанию пользователю доступны данные о модуле (его изображение), данные о проекте (1 – имя проекта, 2 – полный путь к файлу проекта), а также последняя дата редактирования.



## Инструменты для создания и работы с проектами




Данный раздел состоит из следующих элементов:

1. Кнопка создания проекта (перед созданием необходимо указать имя и выбрать путь расположения файла проекта)
2. Графа имени проекта. Следует быть осторожным с типом файла «.sky», он не должен быть удален
3. Кнопка выбора пути расположения проекта. По умолчанию проекты располагаются в папке *C:\Пользователи\«Имя_пользователя»\Документы\ASU_Configurator\Projects*
4. Кнопка переименования проекта. Для переименования необходимо предварительно выбрать проект в списке.
5. Кнопка удаления проекта. Для удаления необходимо предварительно выбрать проект в списке.
6. Кнопка открытия файла проекта. Может пригодиться в тех случаях, когда проект отсутствует в списке. Такое случается, например, после переустановки программы: старые проекты необходимо добавить вручную.
7. Кнопка загрузки прошивки из файла. Кнопка активируется при подключении модуля и позволяет загружать hex-файл напрямую в модуль без необходимости создания нового проекта.

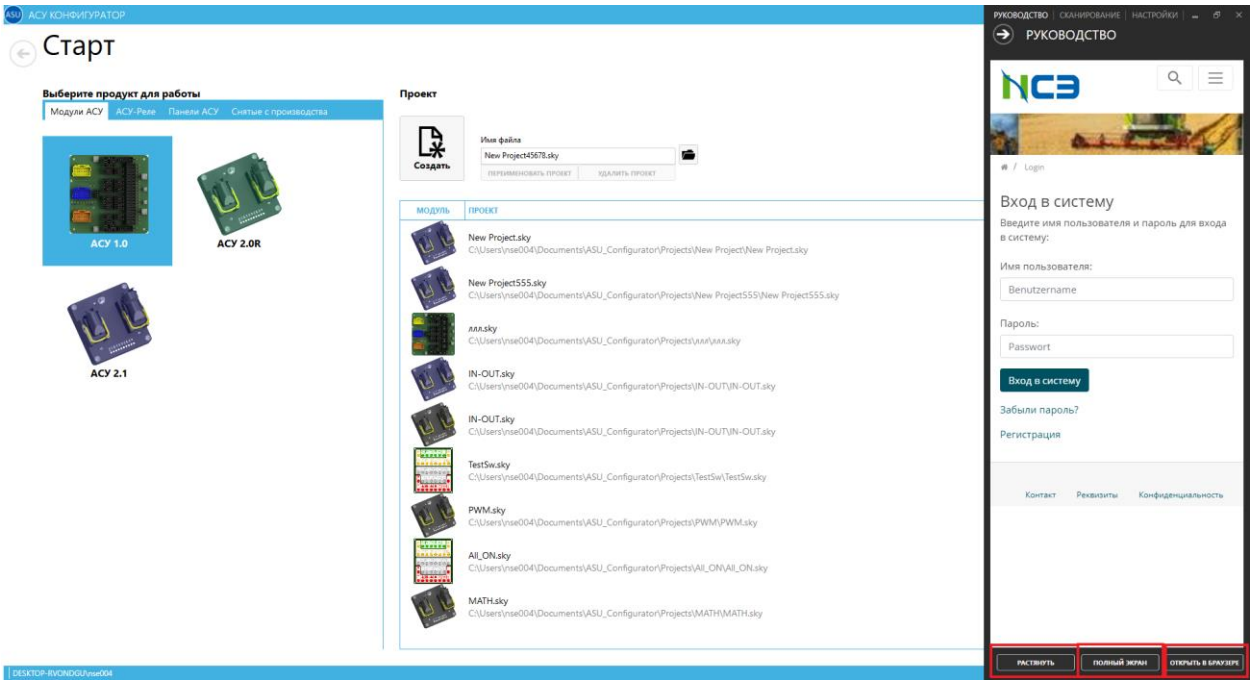
Рядом с кнопками открытия файла и загрузки прошивки из файла расположены кнопки сохранения и компиляции проекта. Эти кнопки активируются после открытия проекта.

Более подробно функционал данной области будет рассмотрен в следующих разделах.

### Меню настроек, сканирование и руководство

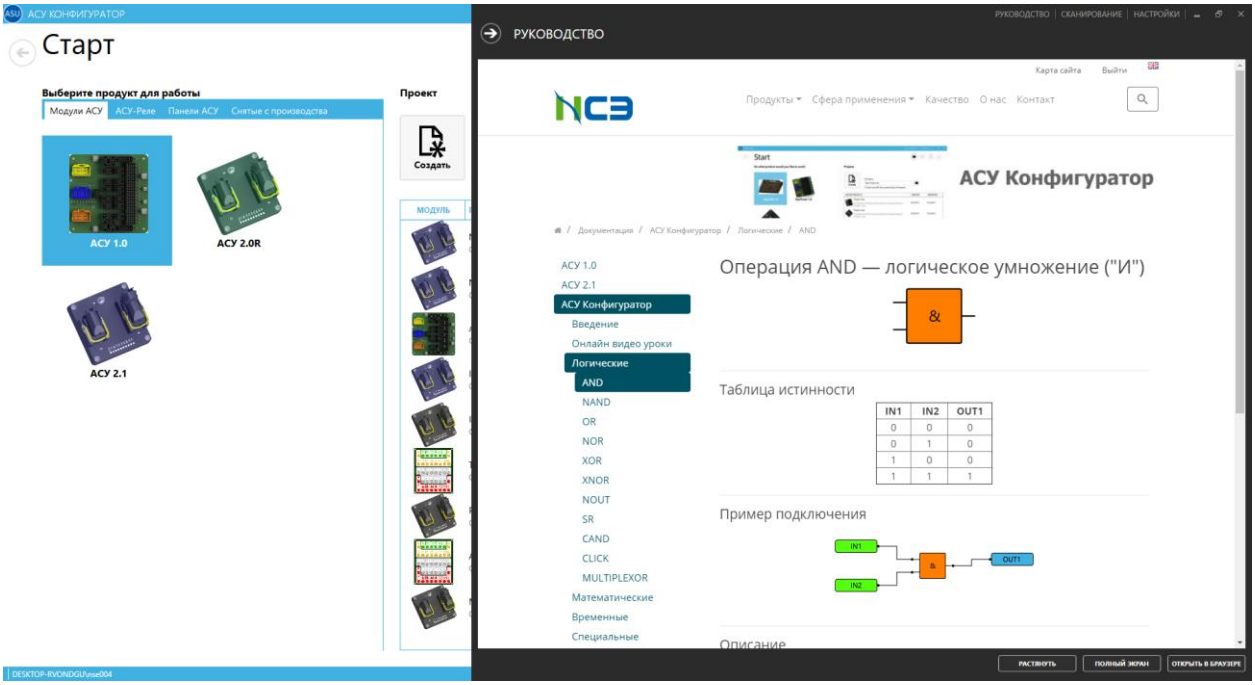
Данная область доступна на всех стадиях работы с программой и включает в себя следующие инструменты:

Руководство – дает возможность пользователю перейти в раздел для просмотра текущего руководства по среде АСУ Конфигуратор. Документ также доступен в папке установки или на сайте компании NSE.

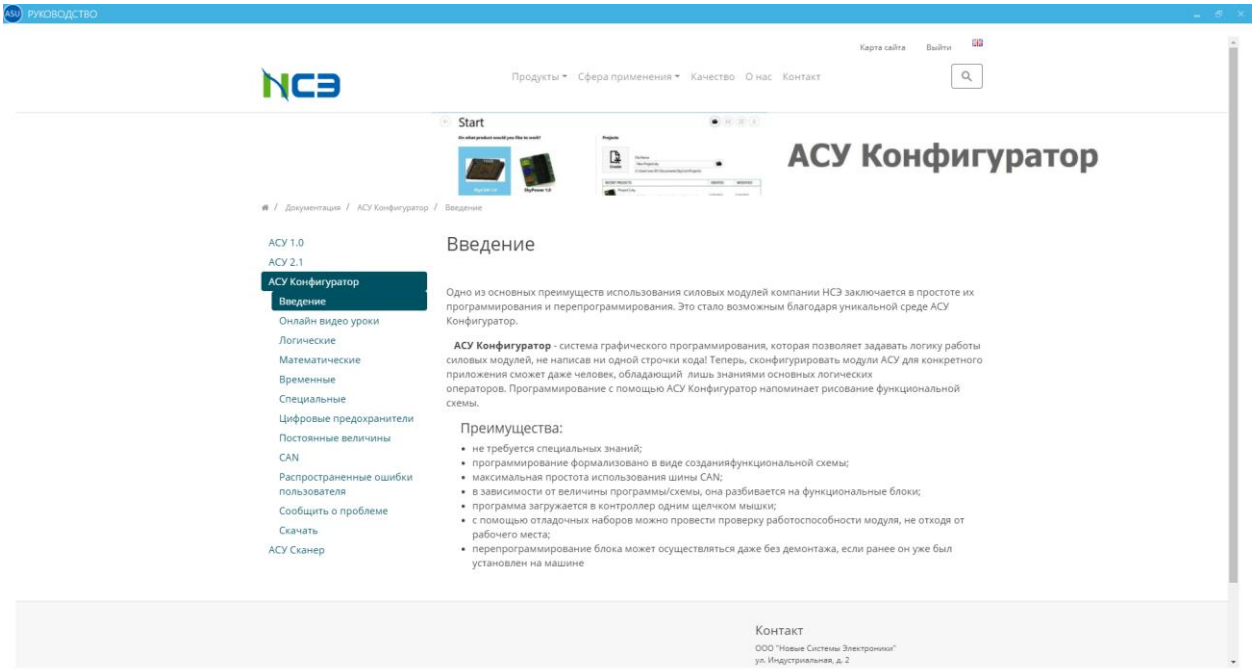



При открытии руководства пользователю доступны 3 кнопки внизу экрана: «растянуть», «полный экран» и «открыть в браузере».

При нажатии на кнопку «растянуть», раздел руководства занимает ¾ рабочего поля. При повторном нажатии размер раздела возвращается к предыдущим параметрам.



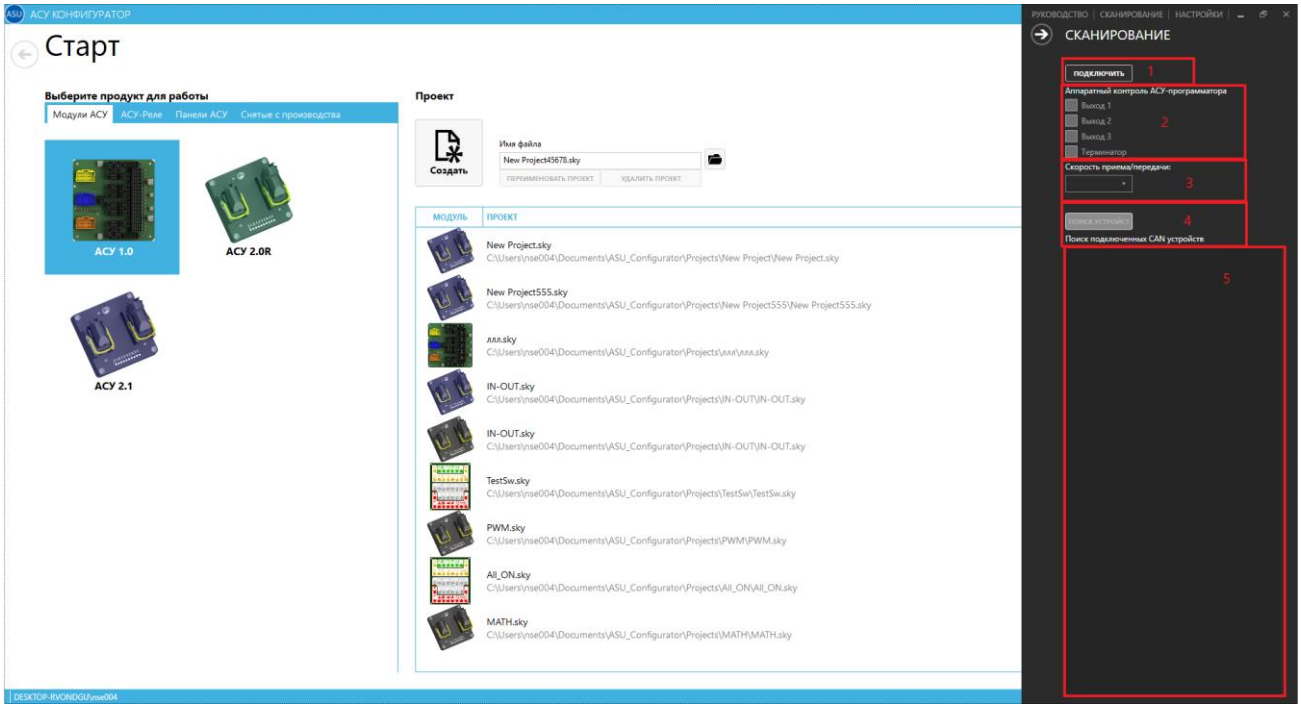

При нажатии кнопки «Полный экран», раздел руководства открывается в отдельном диалоговом окне, которым пользователь может оперировать также, как и другими окнами программы.



При нажатии кнопки «Открыть в браузере», происходит открытие браузера, установленного в параметрах Windows по умолчанию.

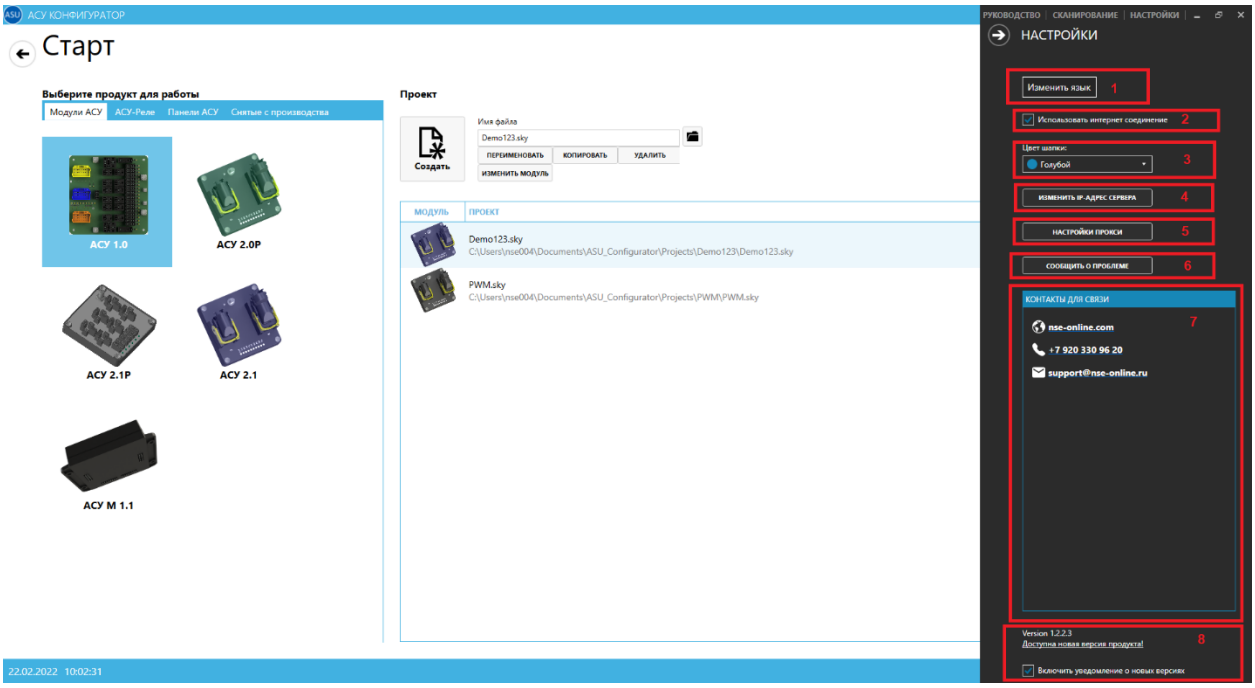
Раздел «Сканирование» необходим для подключения модуля к программной среде и состоит из:

1. Кнопка подключения АСУ-программатора к системе
2. Аппаратный контроль АСУ-программатора (включение/выключение его входов и терминатора)
3. Графа выбора скорости сканирования (скорости работы CAN-интерфейса подключенного модуля, обычно это 250 кб/с)
4. Кнопка поиска устройств. При нажатии АСУ-программатор начинает посылать запрос на подключенные модули в ожидании обратного отклика.
5. Область отображения найденных модулей (показываются ID найденных модулей)

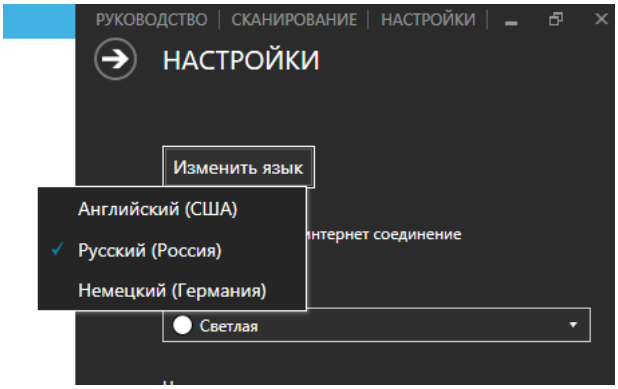


Раздел «Настройки». В разделе доступны следующие инструменты:

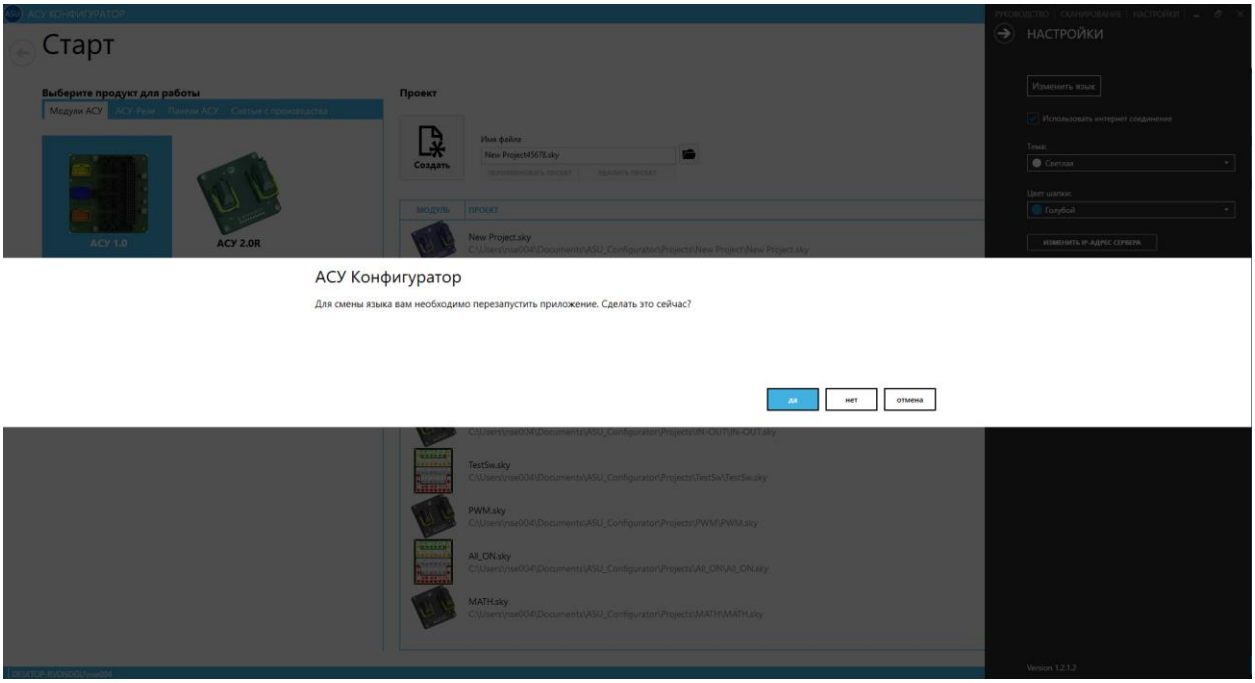
1. Смена языка
2. Включение/отключение Интернет-соединения
3. Выбор цвета шапки
4. Смена IP-адреса сервера
5. Настройки прокси
6. Кнопка для отправки сообщения в техподдержку
7. Контакты для связи по вопросам программирования и настройки
8. Версия программы, ссылка на новую версию программы, если она доступна




Для смены языка необходимо выбрать соответствующий язык в сплывающем меню.



После чего система предложит сменить язык после перезапуска программы.

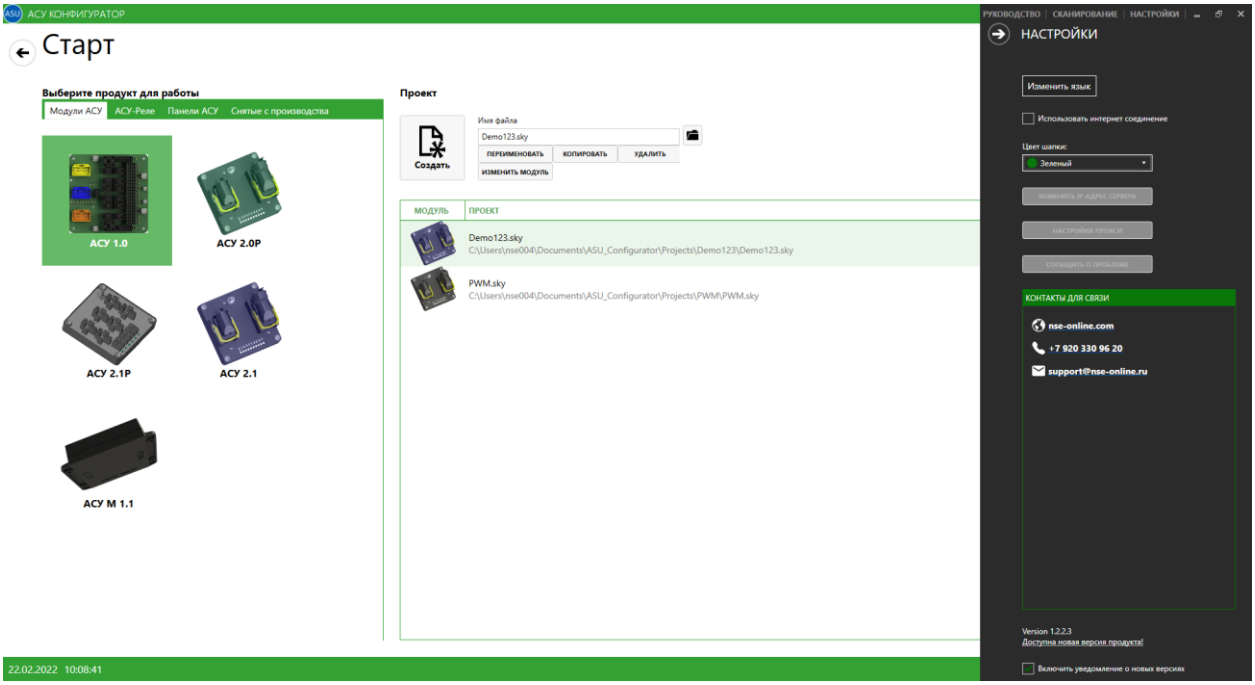



При нажатии кнопки «Да» программа будет автоматически перезагружена с измененными языковыми параметрами.

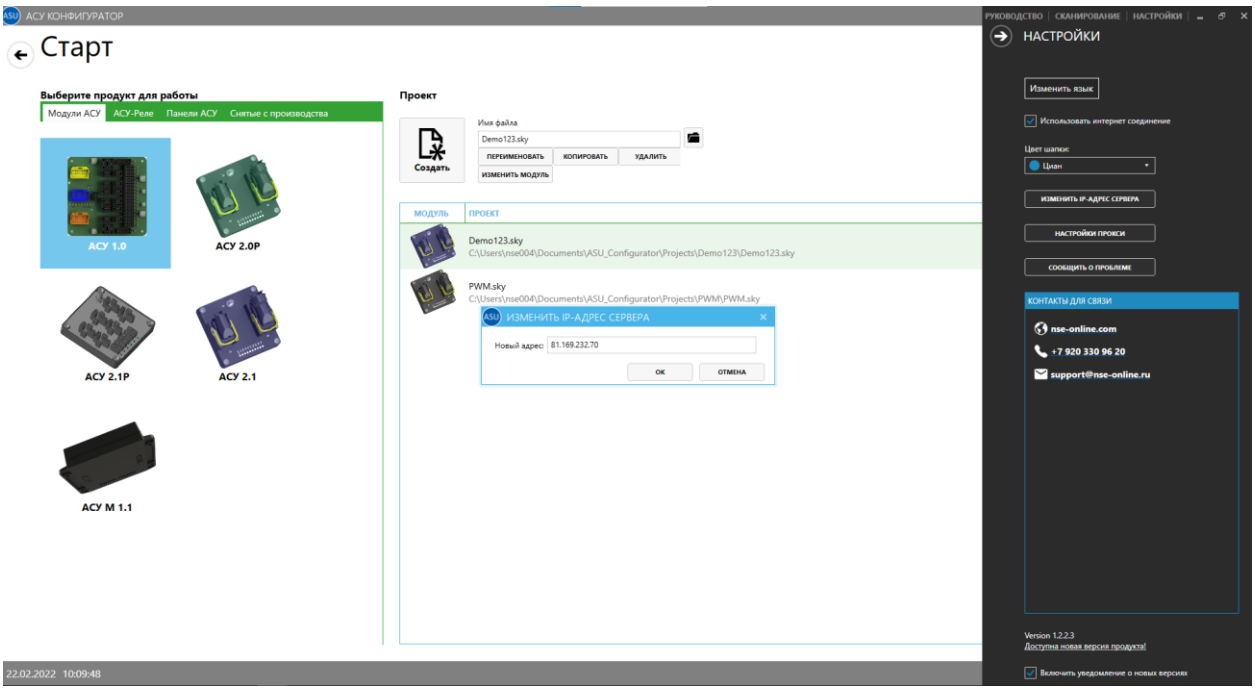
При нажатии «Нет» языковые параметры будут изменены при следующем перезапуске программы.

Включение/отключение Интернет-соединения позволяет пользователю выбрать будут ли проекты компилироваться через сервер компании НСЭ или использовать компилятор, установленный на компьютере пользователя. При использовании внешних серверов необходимо подключение к сети Интернет, что в некоторых случаях бывает невозможно. Использование серверов НСЭ дает гарантии об актуальности создаваемого машинного кода. При отключении Интернет-соединения, кнопки изменения IP-адреса и настроек прокси становятся не активны.

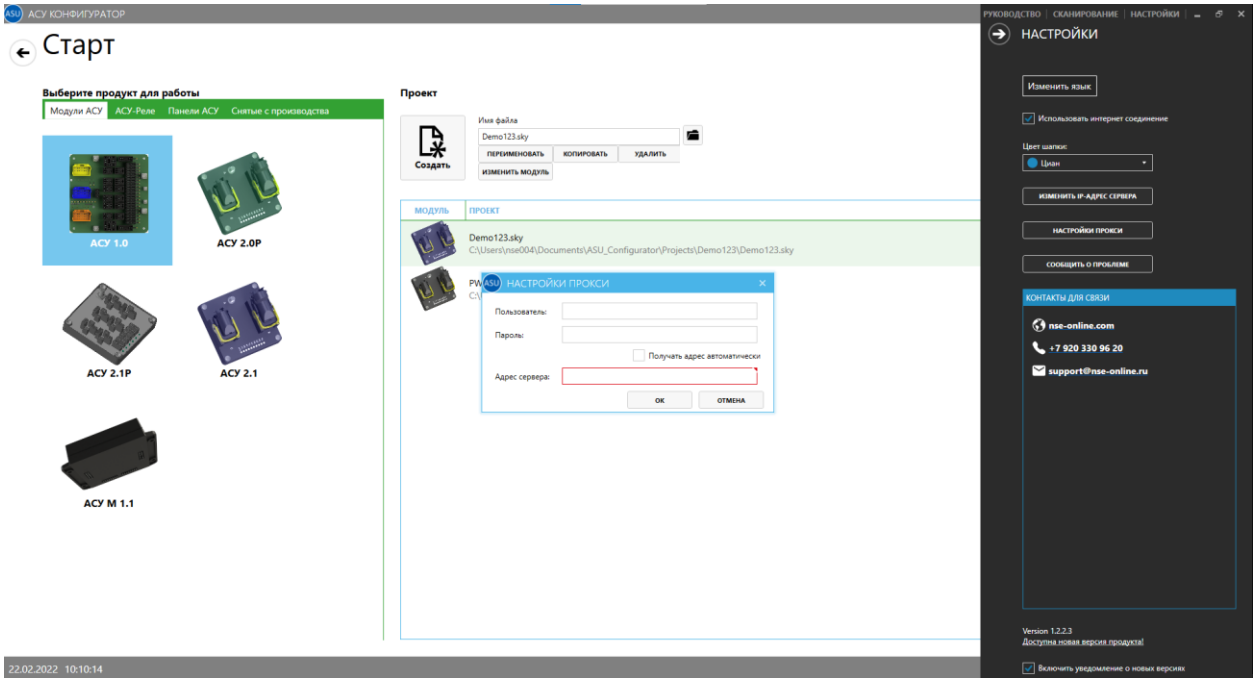
Выбор цвета шапки. Цвет шапки позволяет сделать интерфейс программы более приятным для восприятия.



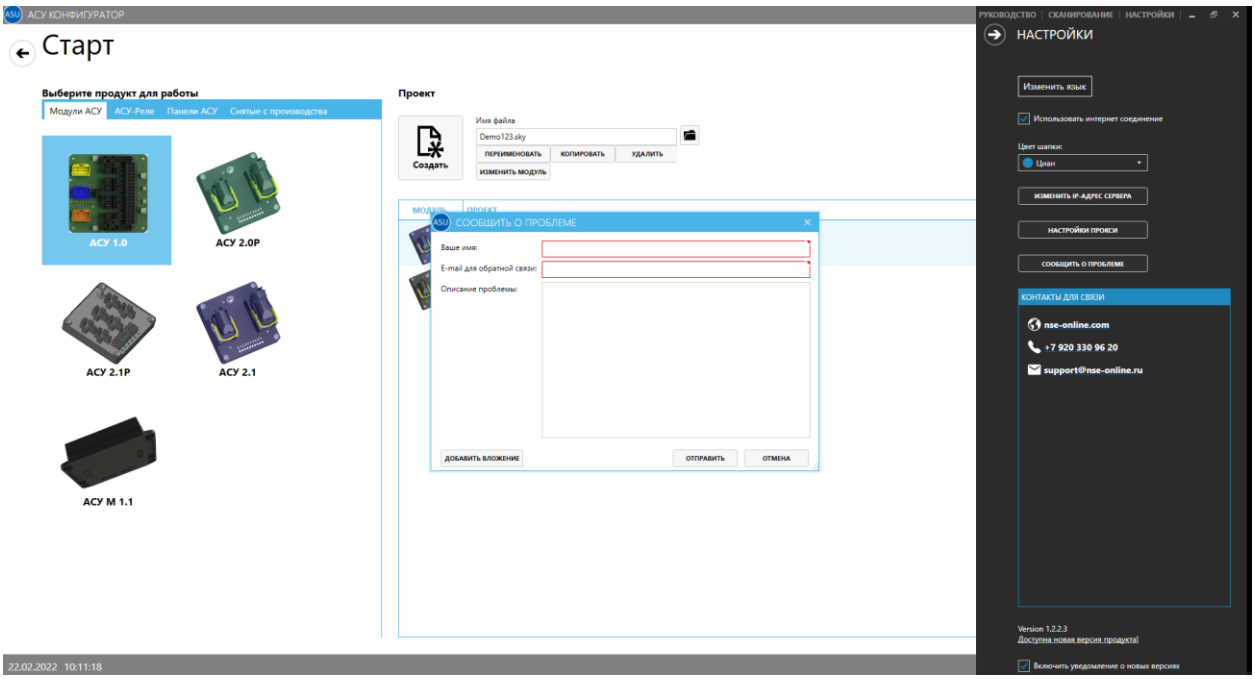

Изменение IP-адреса Данная функция может понадобиться в случае непредвиденной смены IP-адреса сервера НСЭ, в таком случае адрес можно будет изменить вручную без переустановки программы.



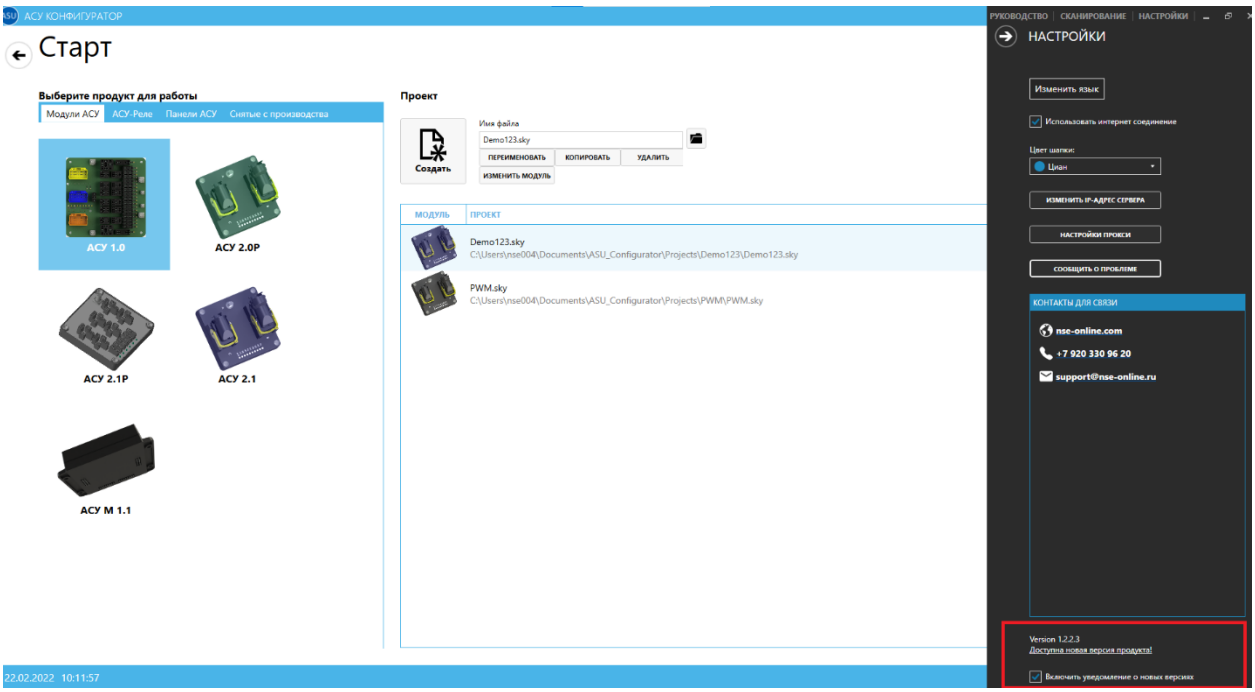
Настройки прокси. Данная функция находится в тестовом режиме. При нажатии на кнопку, пользователю необходимо установить данные о пользователе и адрес прокси-сервера, либо выбрать графу «получать адрес автоматически», в таком случае адрес будет взят из настроек Windows.




Сообщить о проблеме. После нажатия на кнопку «Сообщить о проблеме» появится окно с формой обратной связи, в котором необходимо указать имя, адрес электронной почты, а также описать проблему или предложение. Существует возможность добавления к сообщению файла (кнопка в левом нижнем углу). При нажатии на кнопку «Отправить», на почту НСЭ поступит письмо с заполненными данными.




Версия программы. Внизу раздела настроек отображена версия установленной программы, версия сверяется с данными на сайте компании НСЭ и, в случае несоответствия, предлагает пользователю обновить программу до актуальной. Также в данной графе можно отключить или включить всплывающее при запуске уведомление о наличии новой версии.



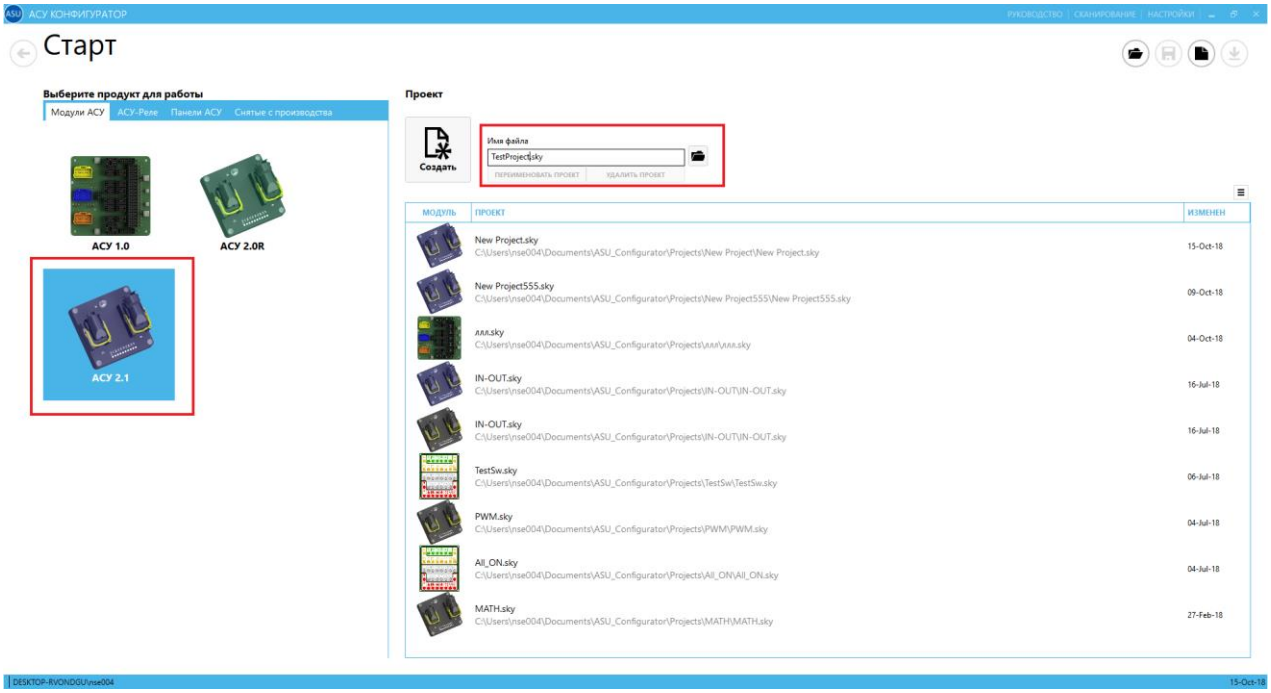
# Создание проекта и настройки рабочего поля

## Создание проекта и функции

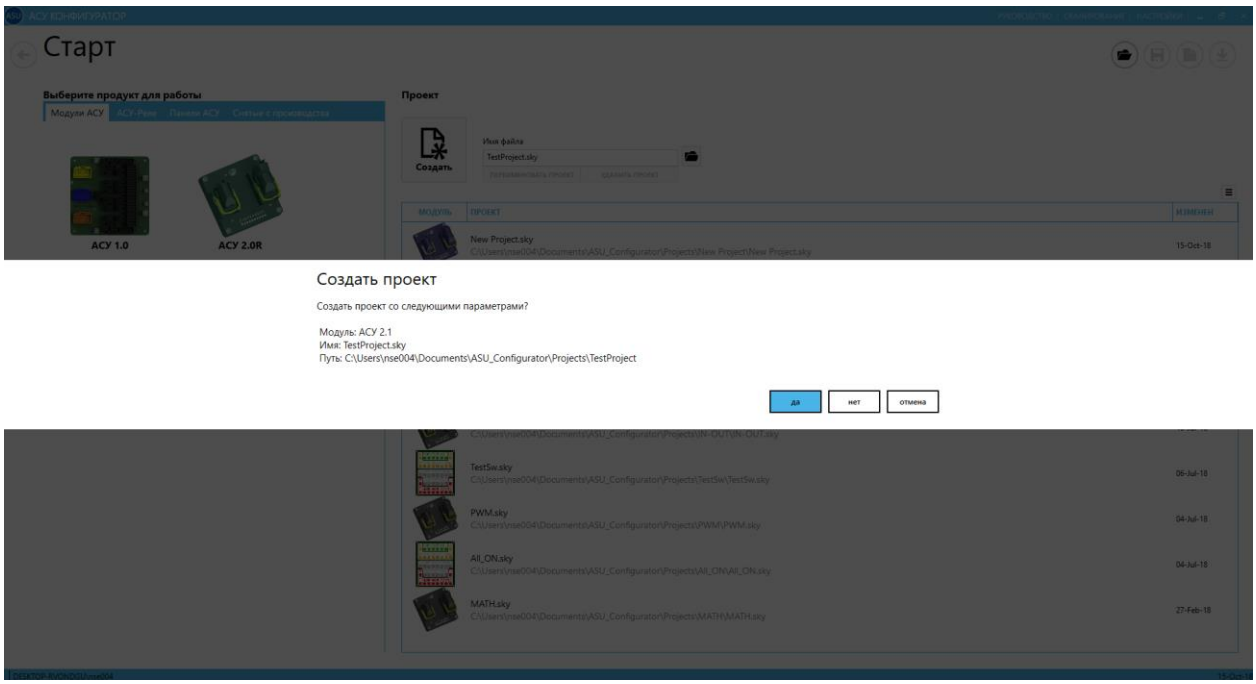
Для создания нового проекта необходимо выполнить следующие действия:

Выбрать модуль, для которого будет создан проект, задать имя проекта, выбрать путь расположения файла проекта или оставить путь по умолчанию и нажать кнопку «Создать».

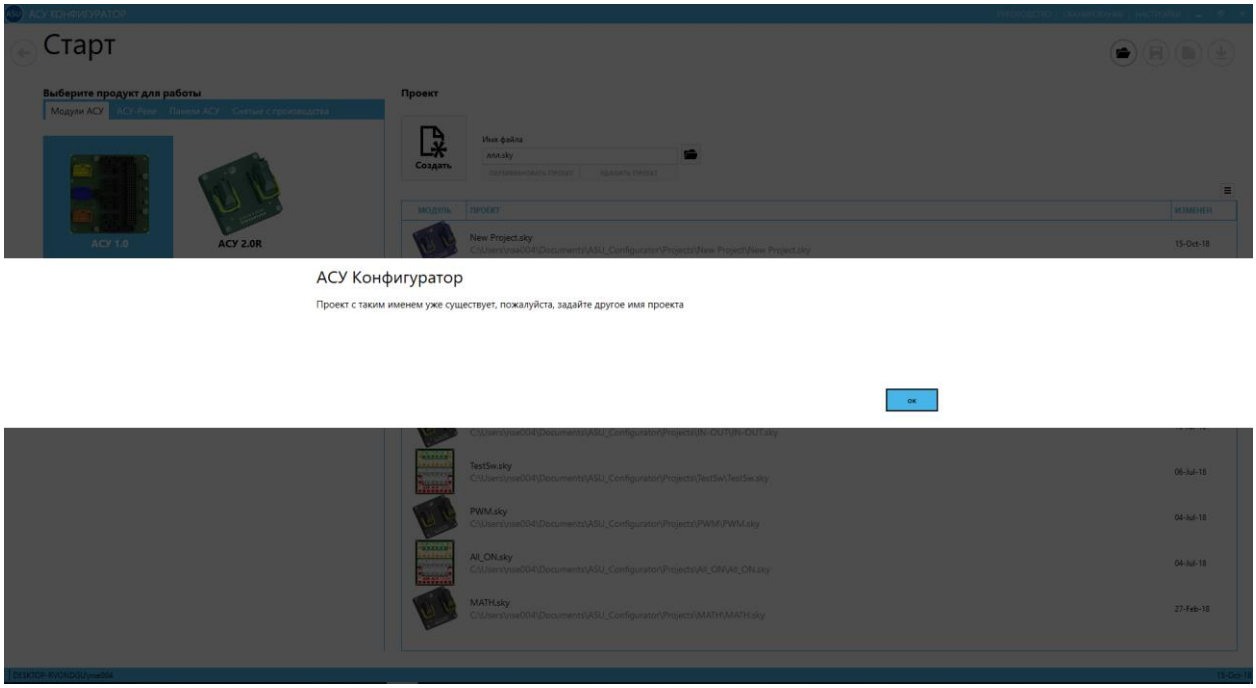
Для примера создадим проект для модуля АСУ 2.1 с именем TestProject




После нажатия кнопки «Создать» появится окно с подтверждением (перед созданием проекта рекомендуется проверить данные).



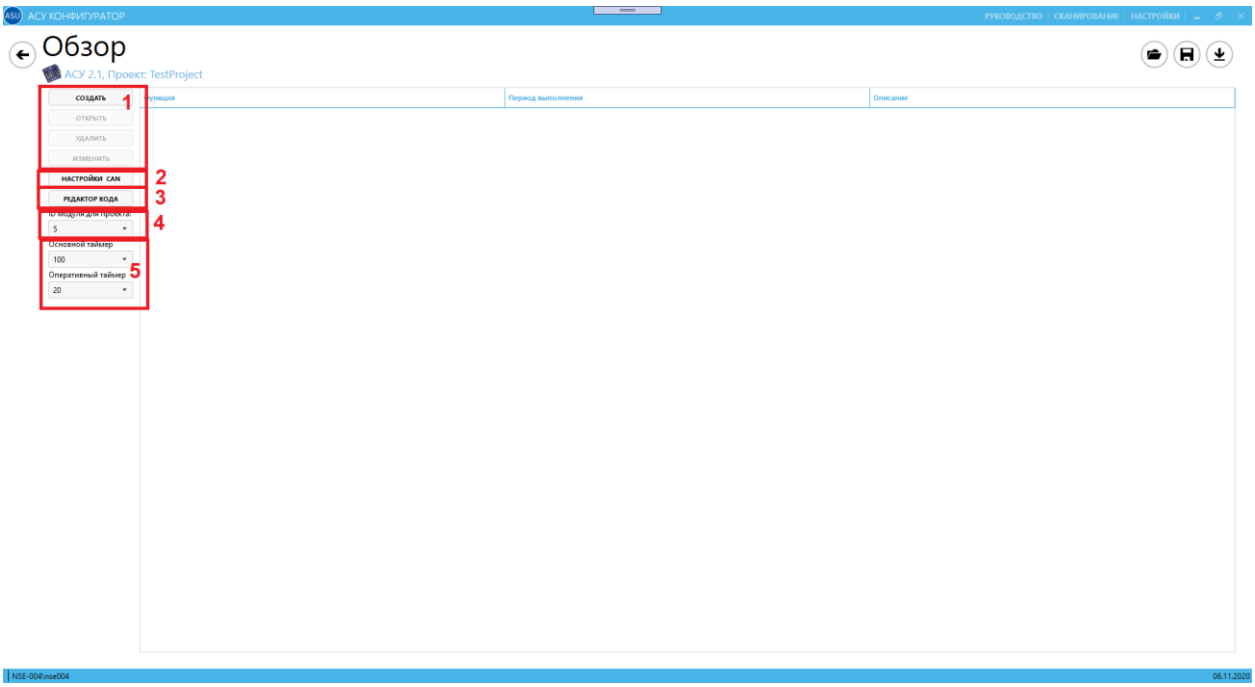

Если проект с таким именем уже существует, появится окно с соответствующим уведомлением.



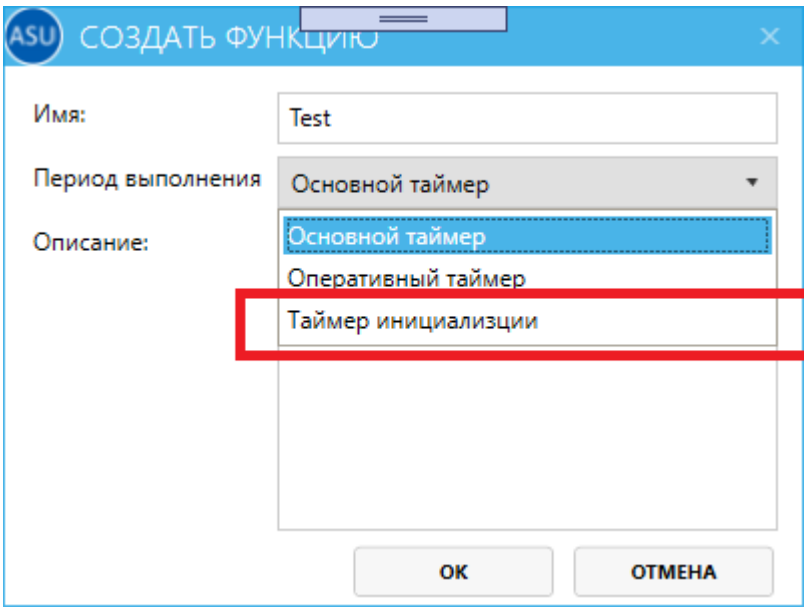
Как только проект создан, его открытие производится автоматически и перед пользователем предстает раздел с функциями проекта.

В данном разделе доступны:

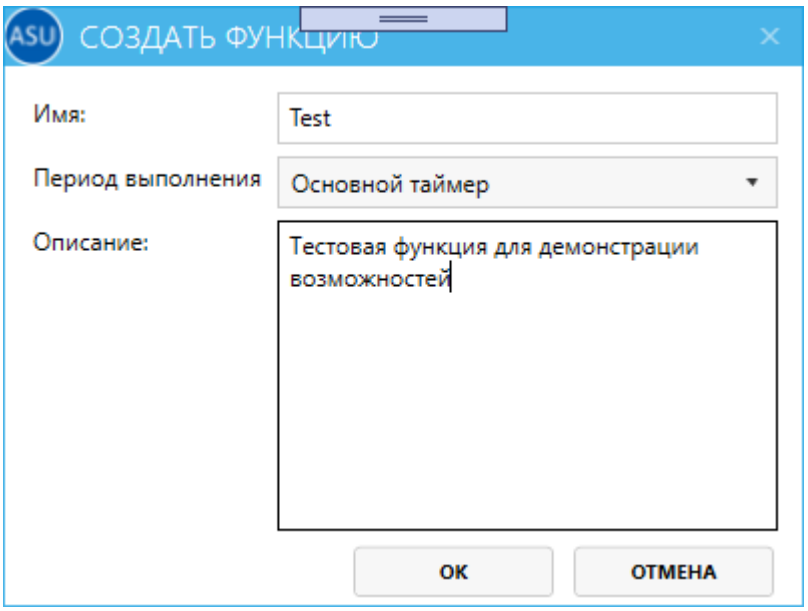
1. Кнопки для работы с функциями
2. Кнопка открытия настроек CAN-сообщений
3. Кнопка открытия редактора кода
4. Графа установки ID модуля для текущего проекта
5. Настройка времени цикла выполнения программы для основного и дополнительного (оперативного) таймеров. По умолчанию установлены значения 100 и 50 мс. Установка времени цикла позволяет увеличить или уменьшить время реакции на изменение состояния входов или CAN сообщений. Однако при изменении данных значений необходимо учесть следующие особенности:
   При установлении слишком маленького значения периода выполнения и большого количества операций и блоков в функции, реальный период выполнения может не совпасть с установленным значением, т.е. для обхода всех блоков микроконтроллеру понадобится больше времени, чем указал пользователь.
   Установленный период не влияет на временные блоки (задержка, импульс, одновибратор и так далее), значения, устанавливаемые на их входе, считаются в секундах.
   В случае если период выполнения меньше периода принимаемых на модуль сообщений проверки состояния CAN (касается *только блоков проверки*), могут возникнуть ложные срабатывания. Подробнее об этом написано в конце раздела «Создание CAN-функций».



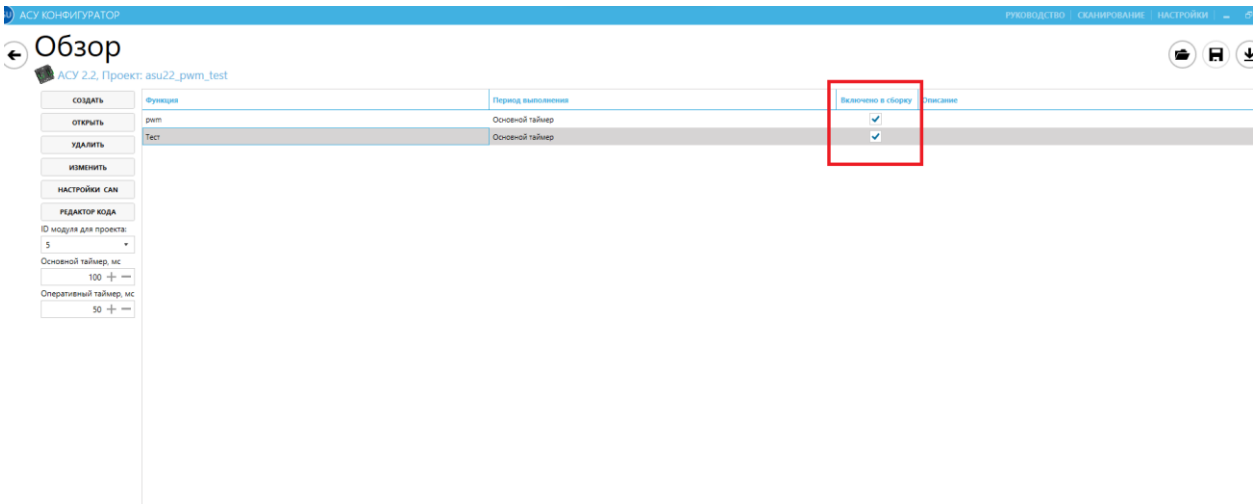

Для дальнейшей работы создадим функцию, для этого нажмем кнопку «Создать» и в диалоговом окне заполним имя и описание функции, а также выберем период выполнения путем выбора таймера. Обратите внимание, что при выборе таймера доступен «Таймер инициализации»: функции или код, заданный для этого таймера, будут выполняться только один раз при старте программы (при подаче питания на модуль).




По умолчанию выбран «Основной таймер», на нем пока что и остановимся.




После создания функции вы увидите секцию «Включено сборку». Диаграммы, созданные внутри функций, во время компиляции конвертируются в код Си. И если вы не хотите включать код конкретной функции в сборку для компиляции, вы можете снять галочку. Это бывает особенно полезно при отладке проекта.

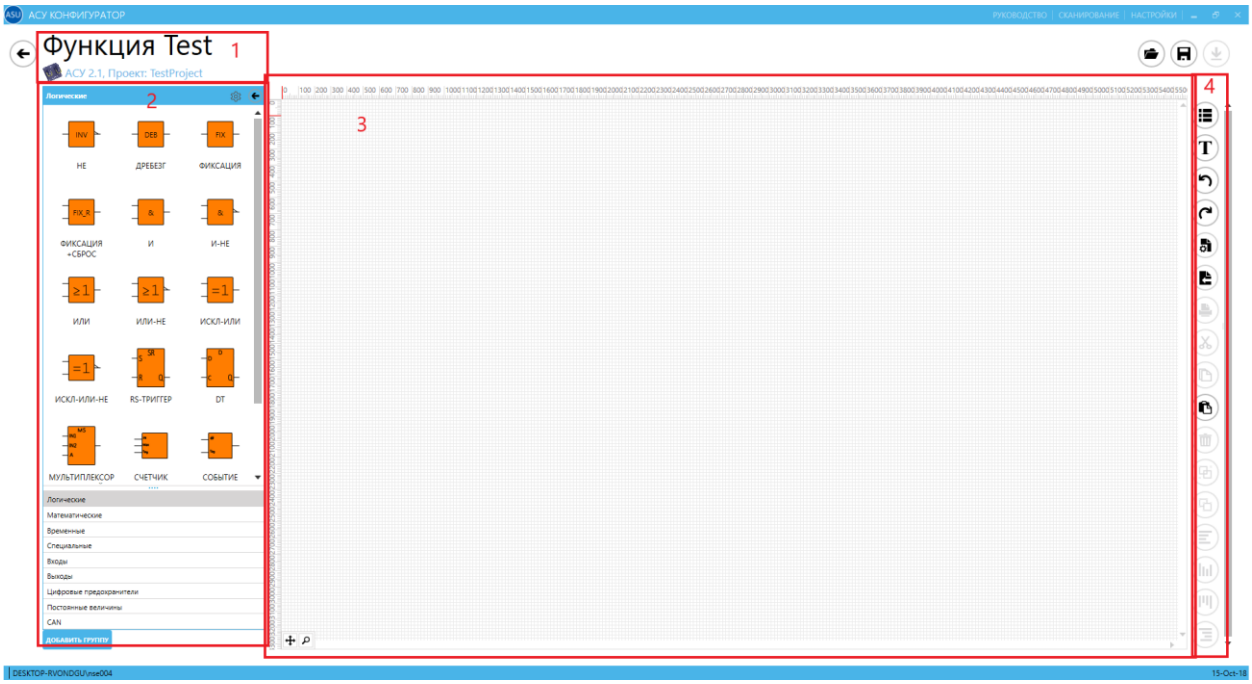



После создания функции, откроем ее двойным щелчком мыши или нажатием на кнопку «Открыть».

## Настройки рабочего поля

После открытия пользователю становится доступно рабочее поле функции, состоящее из следующих разделов:

1. Имя функции и проекта
2. Группы элементов
3. Рабочее поле
4. Кнопки настройки рабочего поля



Рассмотрим кнопки для настройки рабочего поля:

Показать/скрыть редактор кода

Активация текстового указателя

Шаг назад

Шаг вперед

Настройки страницы

Вырезать

Копировать

Вставить

Удалить

Сгруппировать

Разгруппировать

Выровнять по левому краю

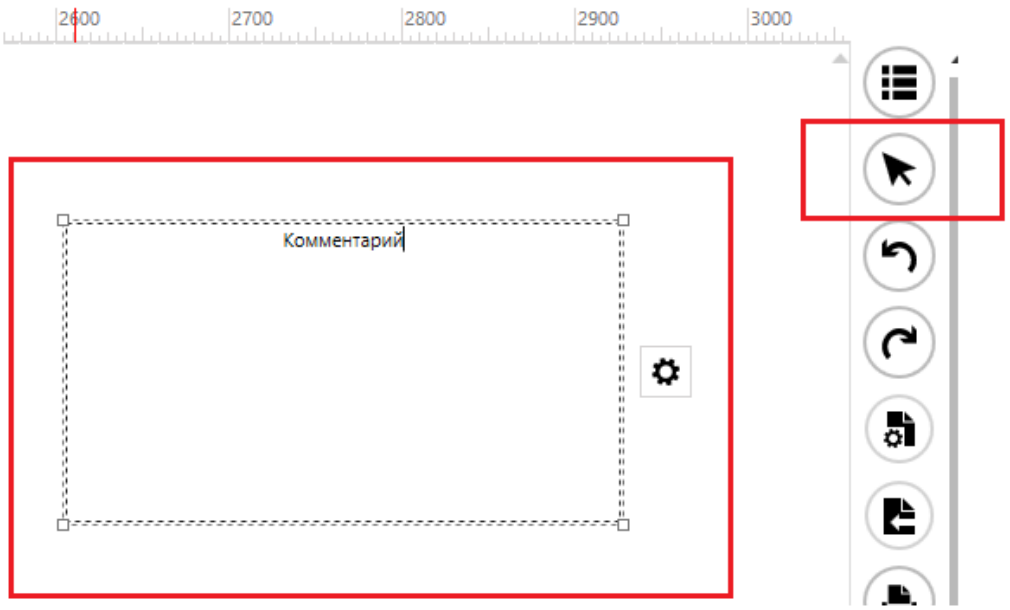
Выровнять по нижней границе

Выровнять по верхней границе

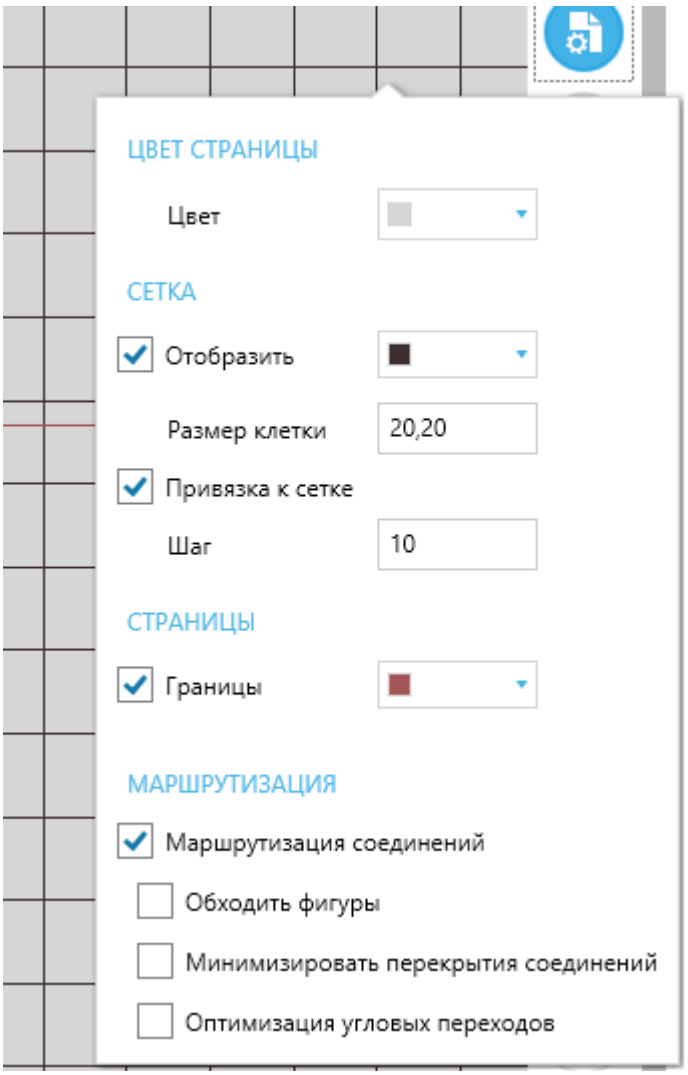
Выровнять по правому краю

Активация текстового указателя. Данная функция предоставляет возможность оставлять текстовые комментарии на рабочем поле. Для этого, после перехода в режим текстового указателя необходимо выделить область на

диаграмме, в которой предполагается оставить комментарий. Чтобы вернуться в режим стандартного указателя, необходимо нажать кнопку с указателем мыши. Вы можете настроить размер шрифта и его цвет путем нажатия на колесико шестеренки возле выделенной области.

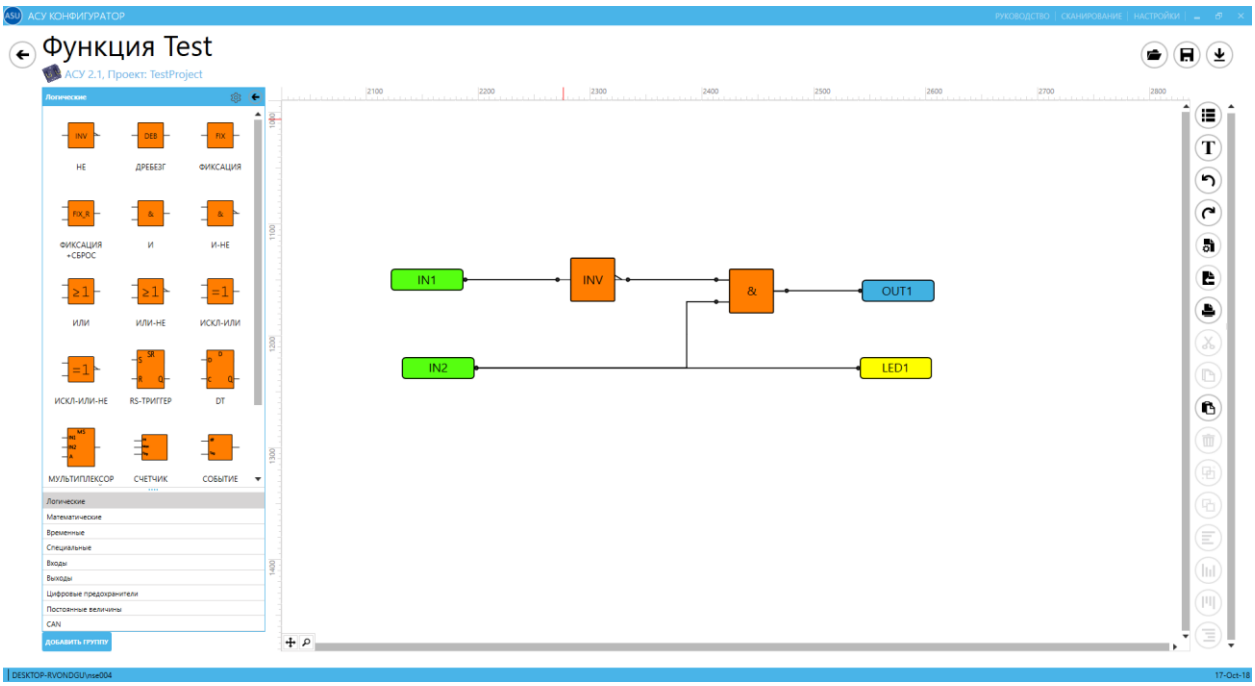



Настройки страницы. Данная функция позволяет настроить цвет фона рабочего поля, размер сетки, ее отображение и цвет, привязку к сетке и шаг перемещения компонентов, а также границы страницы для печати. Обратите также внимание на пункт «Маршрутизация»: данный пункт позволяет настроить поведение линий соединения между элементами. Отключение маршрутизации позволяет сократить время загрузки «насыщенных» схем с большим количеством элементов. Если схема содержит небольшое количество элементов, маршрутизацию рекомендуется оставить для упрощения процесса работы.



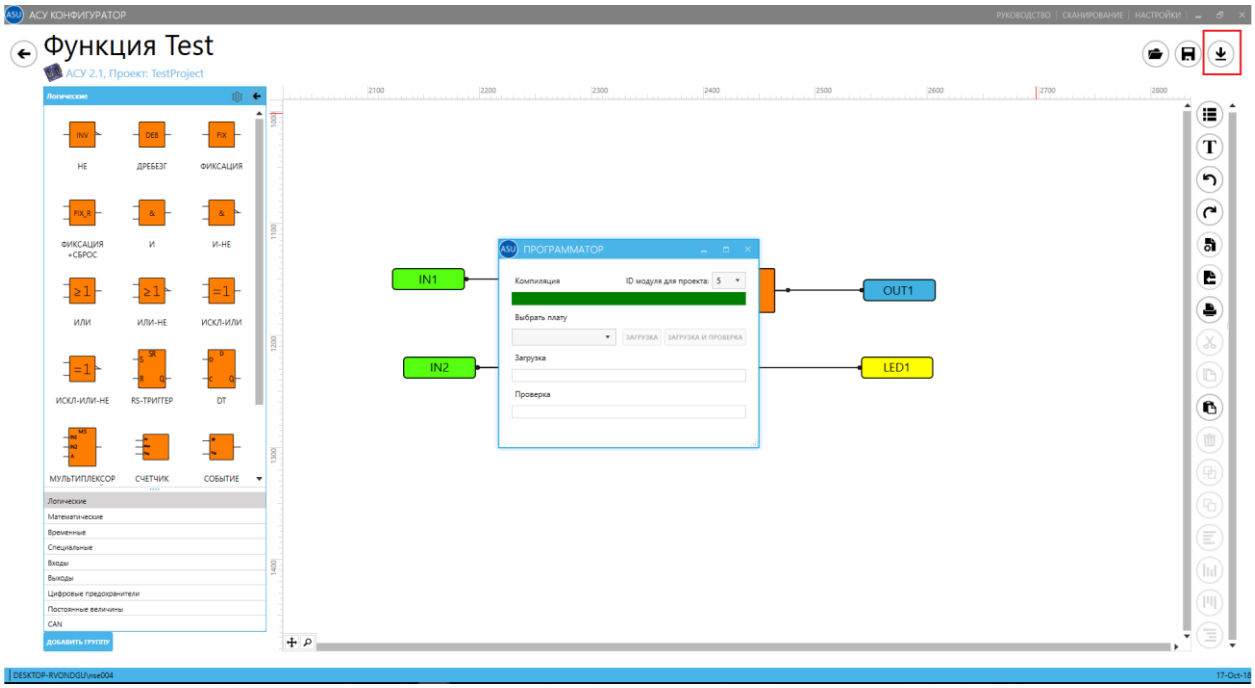

После того как рабочее поле настроено, можно перейти к созданию диаграммы и компиляции проекта.

Создадим простой алгоритм, в соответствии с которым при подаче низкого уровня сигнала на вход IN1 и высокого на IN2, на выход OUT1 поступит высокий сигнал, а на модуле загорится первый светодиод.

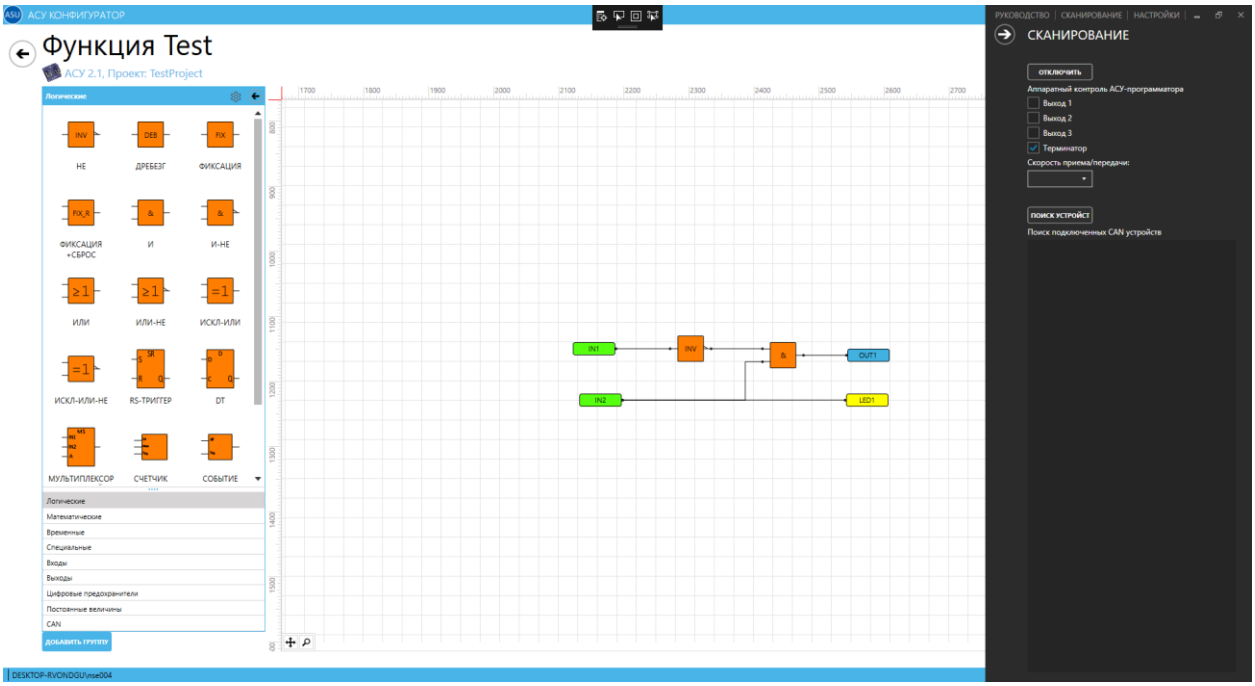



## Компиляция и загрузка проекта

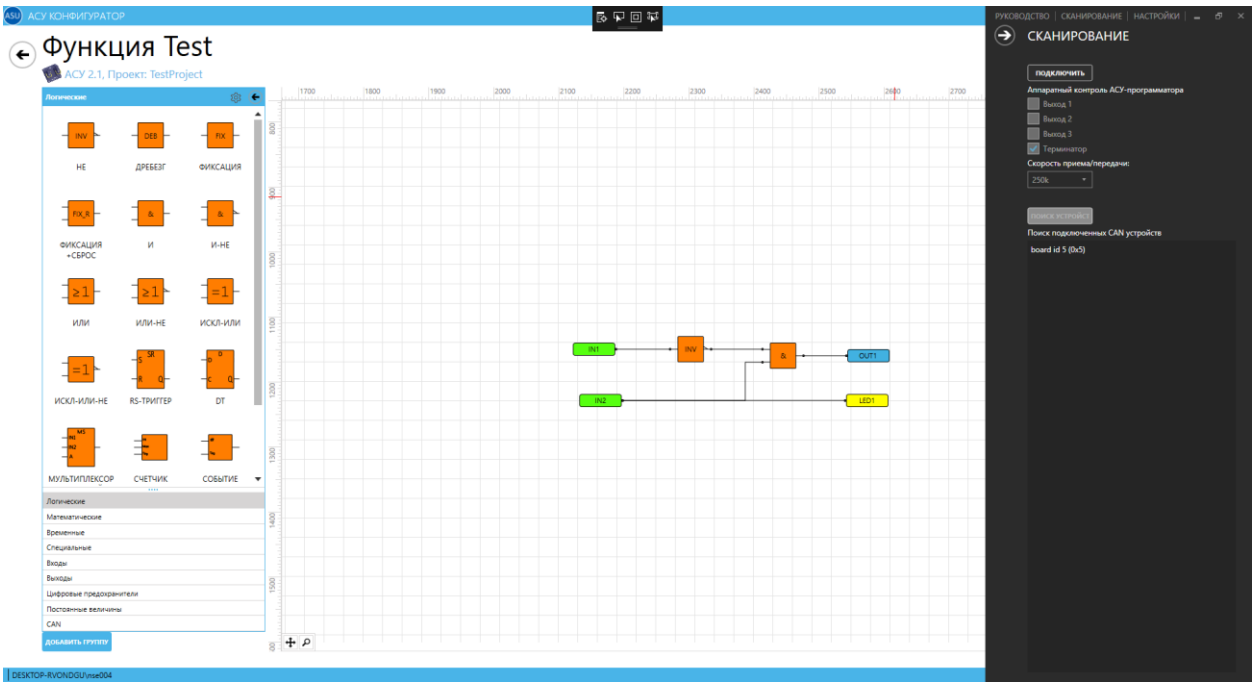
Для компиляции проекта нажмите соответствующую кнопку в правом верхнем углу. Компиляция проекта может занять некоторое время, необходимое для сообщения с сервером и обработки файлов.



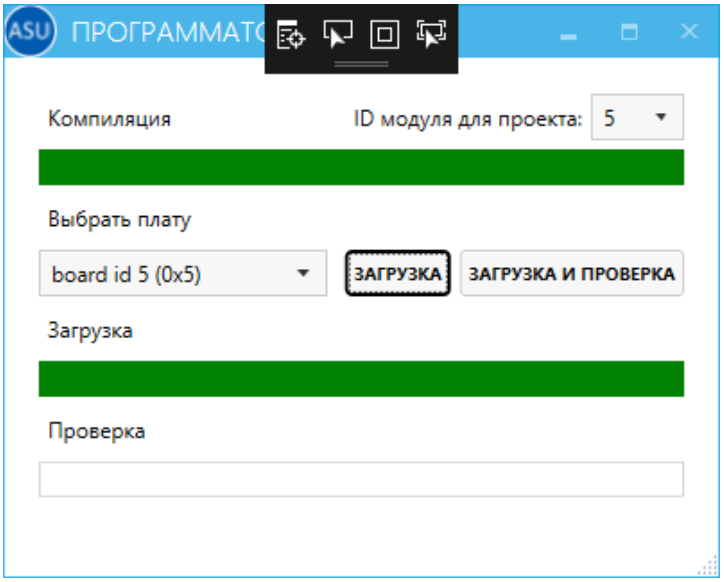
Для загрузки проекта в модуль необходимо выполнить его подключение по схеме: Источник питания – Модуль – АСУ-программатор – Персональный компьютер. Если модулей несколько или линия соединения имеет большую протяженность, может потребоваться включить в цепь резистор-терминатор 120 Ом. Как только подключение выполнено, зайдите в раздел «Сканирование» и нажмите кнопку «Подключить». Выставите Терминатор, в АСУ-программаторе должен раздаться характерный щелчок.




Далее выберите скорость работы модуля (по умолчанию 250) и нажмите «Поиск устройств». Если все было проделано верно, в поле ниже отобразится список ID найденных устройств. В случае, если устройство не найдено, проверьте правильность его подключения, а также выбранную скорость и повторите попытку. Если повторные попытки не приносят результата, воспользуйтесь программой «АСУ-сканер», где аналогично выполните подключение и выберите скорость работы.




После подключения, нажмите кнопку компиляции и в открывшемся окне «Загрузка» или «Загрузка и проверка»



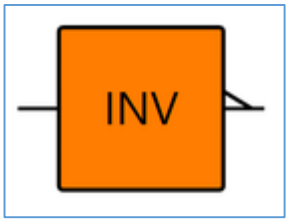

«Загрузка и проверка» позволяет после загрузки программного кода считать его с модуля и сравнить с исходным.

Обратите внимание, что при загрузке программы доступна возможность смены ID модуля, путем задания «ID модуля для проекта», однако после этого модуль необходимо будет перезагрузить для начала работы кода.

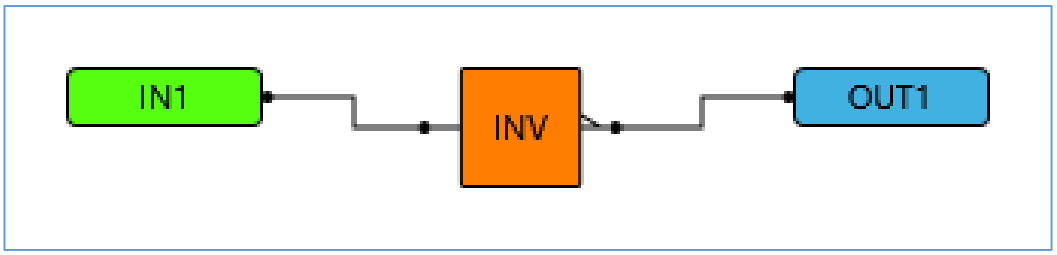
# Группы компонентов и их описание

## Логические

НЕ— логическое отрицание(инверсия)



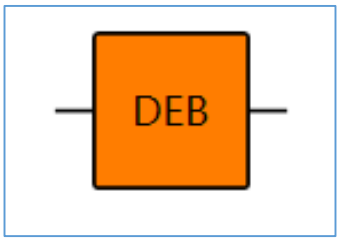

Пример подключения



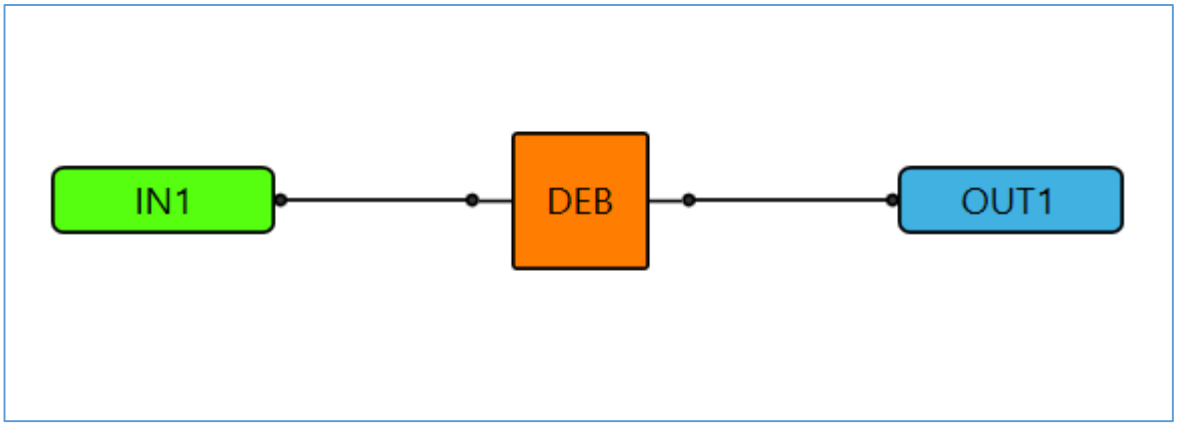

Описание

Простая инверсия сигнала. При подаче высокого уровня сигнала на вход IN1, на выходе OUT1 будет присутствовать низкий уровень и наоборот.

ДРЕБЕЗГ — защита от дребезга



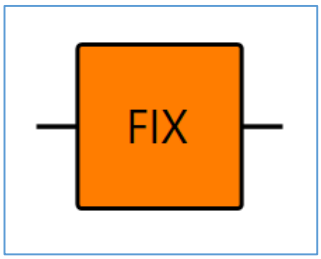

Пример подключения



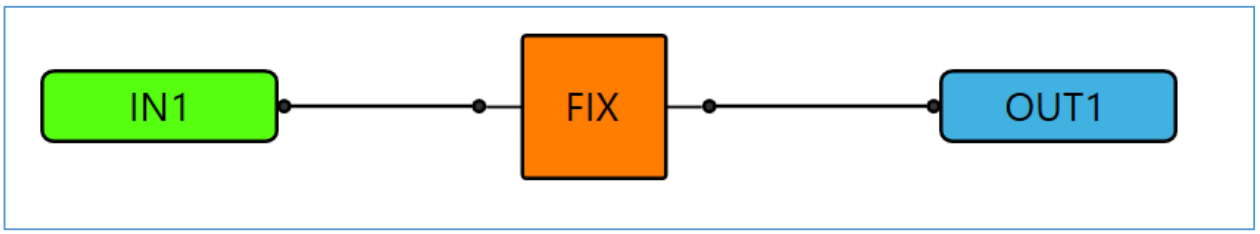

Описание

Компонент защиты от дребезга. Сигнал на выходе соответствует сигналу на входе. Принцип работы заключается в установки задержки в 1мс.

ФИКСАЦИЯ – фиксация сигнала на выходе



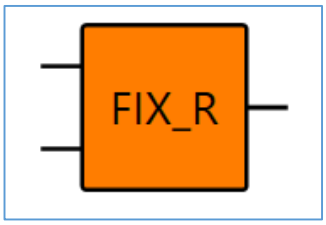

Пример подключения



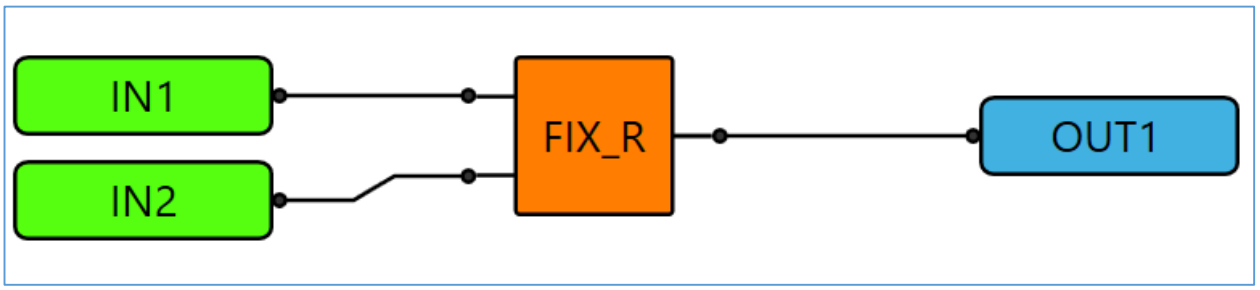

Описание

При подаче фронта сигнала на IN1, на выходе OUT1 появляется высокий уровень сигнала и сохраняется до тех пор, пока на IN1 не придет очередной фронт.

ФИКСАЦИЯ+СБРОС — фиксация + сброс сигнала на выходе




Пример подключения




Описание

При подаче фронта сигнала на IN1, на выходе OUT1 появляется высокий сигнал и сохраняется до тех пор, пока на IN1 или IN2 не придет очередной фронт сигнала.

И — логическое умножение

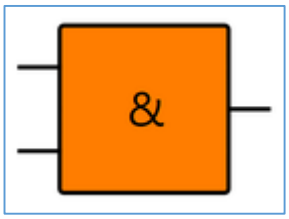
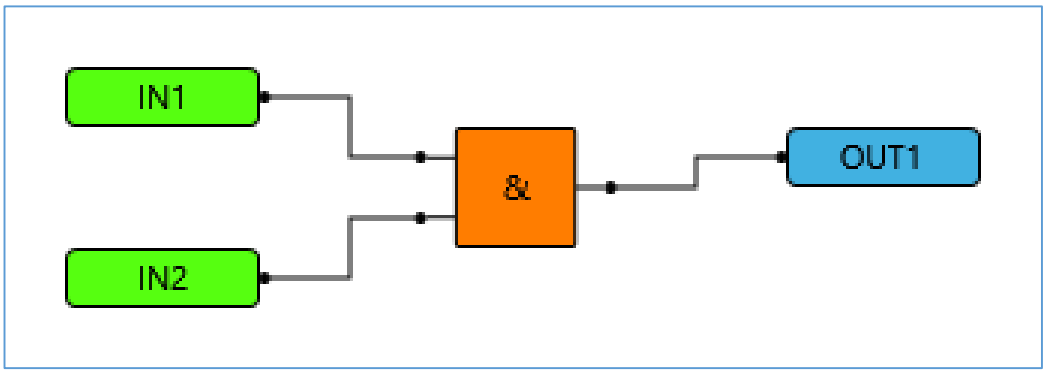



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Пример подключения




Описание

Результат логического умножения (конъюнкции) является истинным только в том случае, когда истинны оба входящих в него простых выражения. Иными словами, при подаче высокого сигнала на входы IN1 и IN2, на выходе OUT1 будет высокий сигнал, в остальных случаях-низкий.

И-НЕ— логическое умножение с инверсией

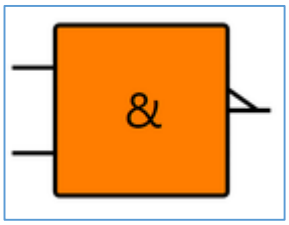
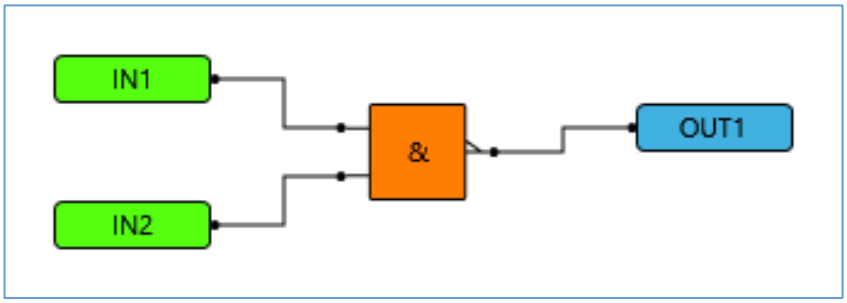



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Пример подключения




Описание

Результат логического умножения (конъюнкции) с инверсией является истинным во всех случаях, кроме случая, когда истинны оба входящих в него простых выражения. Иными словами, при подаче высокого сигнала на входы IN1 и IN2, на выходе OUT1 будет низкий сигнал, в остальных случаях-высокий.

ИЛИ— логическое сложение

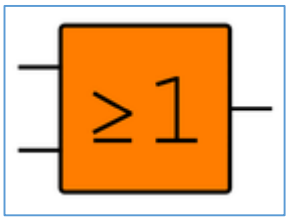
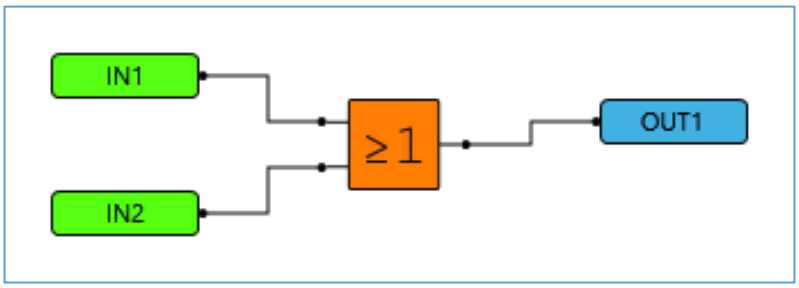



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Пример подключения




Описание

Результат логического сложения (дизъюнкции) является истинным, если хотя бы одно из простых логических входных выражений истинно и ложен, если оба простых логических выражения ложны. Иными словами, при подаче сигнала на входы IN1 и IN2, на выходе OUT1 будет низкий сигнал только в том случае, когда на обоих входах будет низкий сигнал.

ИЛИ-НЕ— логическое сложение с инверсией

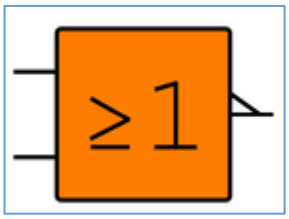
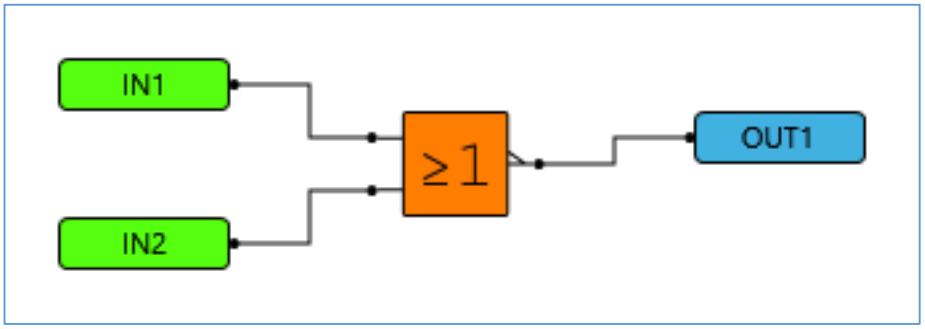



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Пример подключения




Описание

Результат логического сложения с инверсией является ложным, если хотя бы одно из простых логических входных выражений истинно и истинен, если оба простых логических выражения ложны. Иными словами, при подаче сигнала на входы IN1 и IN2, на выходе OUT1 будет высокий сигнал только в том случае, когда на обоих входах будет низкий сигнал.

ИСКЛ-ИЛИ— исключающее ИЛИ

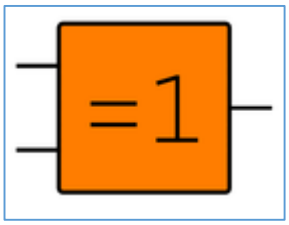
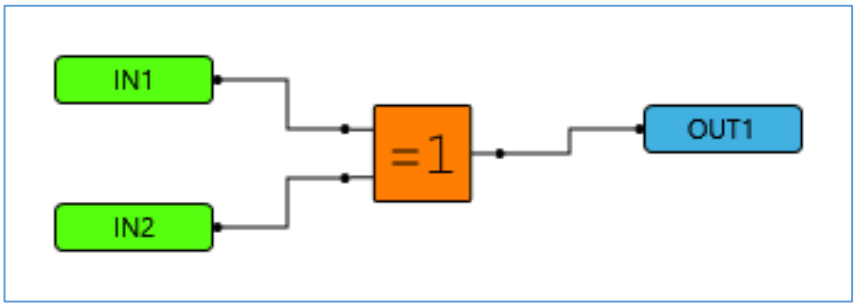



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Пример подключения




Описание

Результат исключающего «или» (строгой дизъюнкции) истинен, если одно из простых логических входных выражений истинно (но не оба сразу) и ложен, если оба простых логических выражения ложны или истины. Иными словами, при подаче сигнала на входы IN1 и IN2, на выходе OUT1 будет высокий сигнал только в том случае, когда только на одном из входов присутствует высокий сигнал.

ИСКЛ-ИЛИ-НЕ—исключающее ИЛИ с инверсией

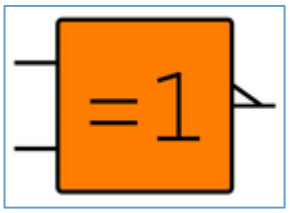
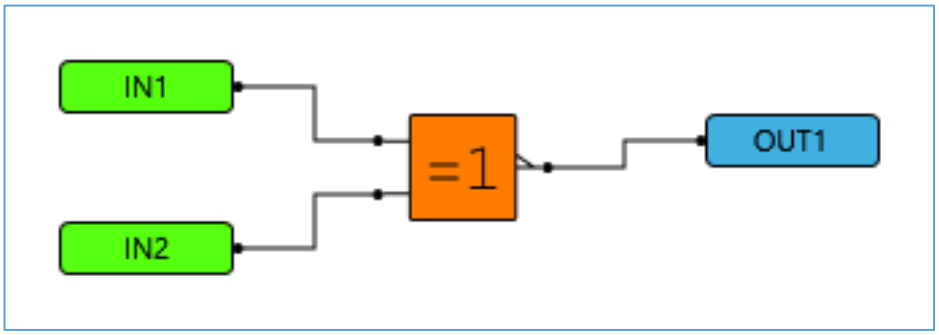



Таблица истинности

| IN1 | IN2 | OUT1 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Пример подключения




Описание

Результат исключающего «или» (строгой дизъюнкции) с инверсией ложен, если одно из простых логических входных выражений истинно (но не оба сразу) и истинен, если оба простых логических выражения ложны или истины. Иными словами, при подаче сигнала на входы IN1 и IN2, на выходе OUT1 будет низкий сигнал только в том случае, когда только на одном из входов присутствует высокий сигнал.

RS-ТРИГГЕР

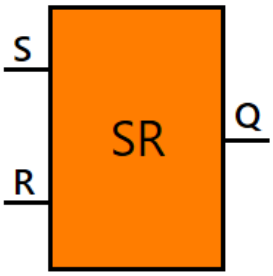
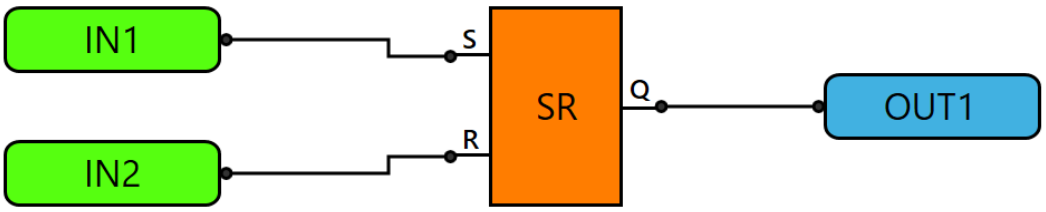



Таблица истинности

| R | S | Q(t) | Q(t+1) | Пояснения |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Режим хранения информации R=S=0 |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 1 | Режим установки единицы S=1 |
| 0 | 1 | 1 | 1 | |
| 0 | 0 | 0 | 0 | Режим записи нуля R=1 |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | * | Запрещенная комбинация R=S=1 |
| 1 | 1 | 1 | * | |

Пример подключения




Описание

Компонент RS выполняет логическую операцию, результат которой истинен, если логическое выражение, поданное на первый вход истинно и сохраняет свое значение, пока на второй вход не поступит истинное выражение, в остальных случаях сигнал ложен. Иными словами, при подаче высокого сигнала на вход IN1, на выходе OUT1 будет высокий сигнал до тех пор, пока на вход IN2 не поступит высокий сигнал

D-ТРИГГЕР

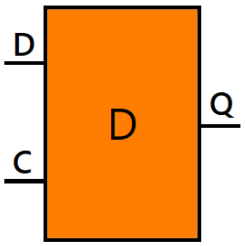
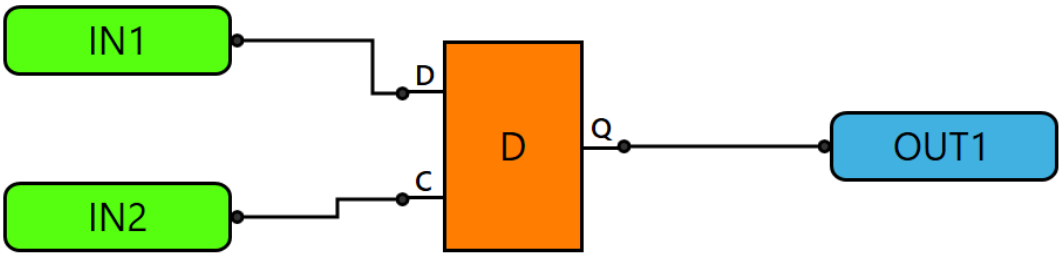



Таблица истинности

| C | Q(t) | D | Q(t+1) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Пример подключения




Описание

Выход компонента будет находиться в устойчивом положении в том случае, если на C=0. В этом случае импульсы, подающиеся на информационный D-вход, никак не влияют на компонент и выходной импульс определяется записанным ранее значением. Если C=1, то выходной сигнал будет зависеть от того, какой сигнал подан на D-вход. Если D=1, то на выходе будет 1, если D=0, то на выходе будет 0.

МУЛЬТИПЛЕКСОР ЦИФРОВОЙ

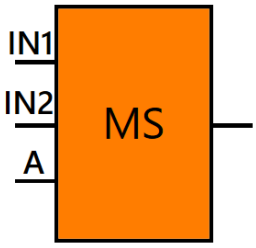
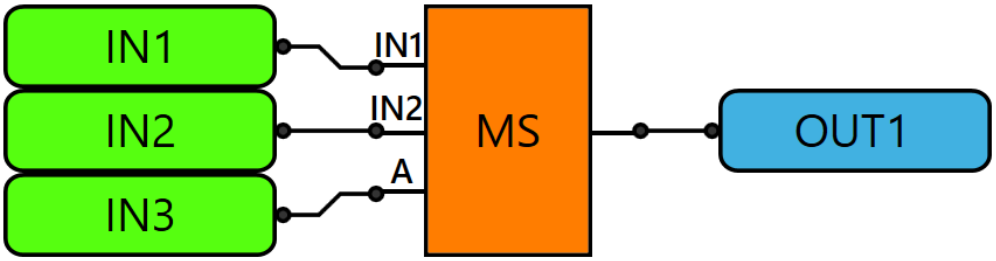



Таблица истинности

| A | IN1 | IN2 | OUT |
|---|---|---|---|
| 0 | 0 | - | 0 |
| 0 | 1 | - | 1 |
| 1 | - | 0 | 0 |
| 1 | - | 1 | 1 |

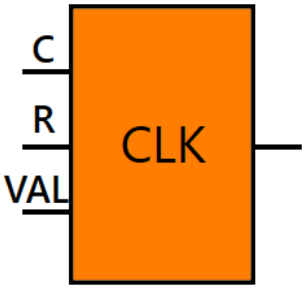
Пример подключения



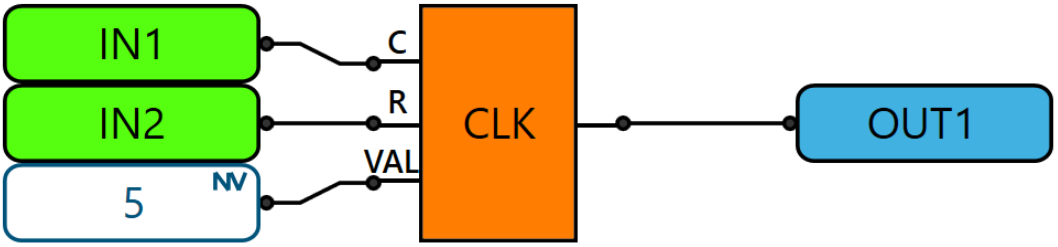

Описание

Компонент является простым двухвходовым мультиплексором, где A-управляющий вход. При наличии высокого сигнала на входе A, выход компонента будет дублировать сигнал на входе IN1, при подаче низкого сигнала на вход A, выход компонента будет дублировать сигнал на входе IN2.

СЧЕТЧИК




Пример подключения

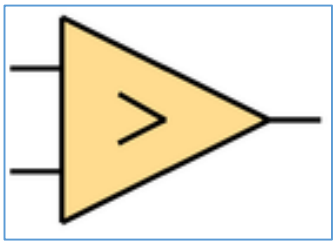



Описание

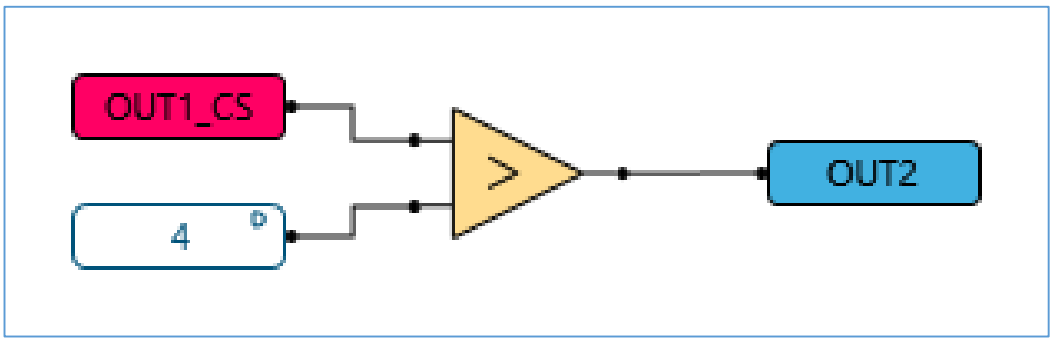
Компонент выполняет подсчет фронтов импульсов на входе C. Вход R выполняет сброс счетчика. Вход VAL задает номер импульса, при котором на выходе будет установлен высокий уровень сигнала.

## Математические

### БОЛЬШЕ



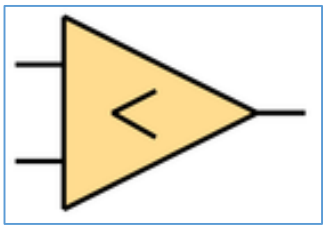
Пример подключения



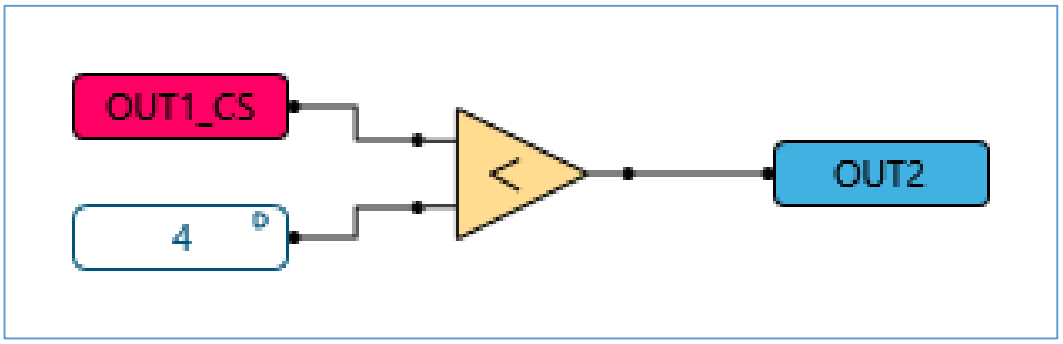

Описание

Компонент выполняет операцию сравнения численных величин или аналоговых сигналов. Если рассматривать пример подключения, то в том случае, когда уровень тока на выходе OUT1 больше 4 ампер, на выходе OUT2 будет присутствовать высокий сигнал.

МЕНЬШЕ



Пример подключения

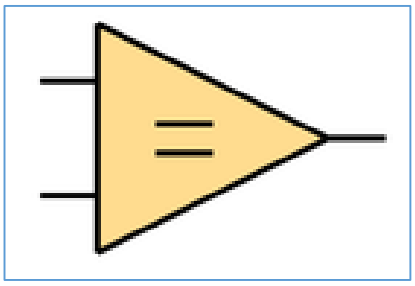



---

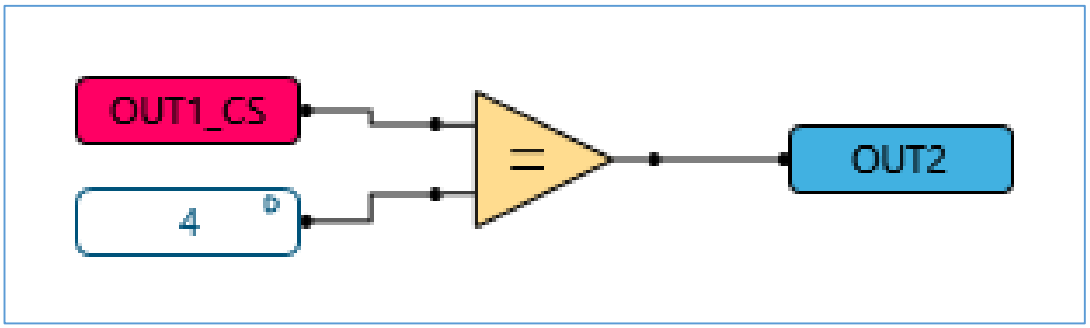
Описание

Компонент выполняет операцию сравнения численных величин или аналоговых сигналов. Если рассматривать пример подключения, то в том случае, когда уровень тока на выходе OUT1 меньше 4 ампер, на выходе OUT2 будет присутствовать высокий сигнал.

РАВНО



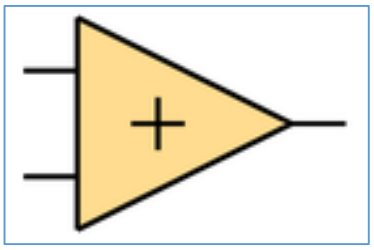
Пример подключения



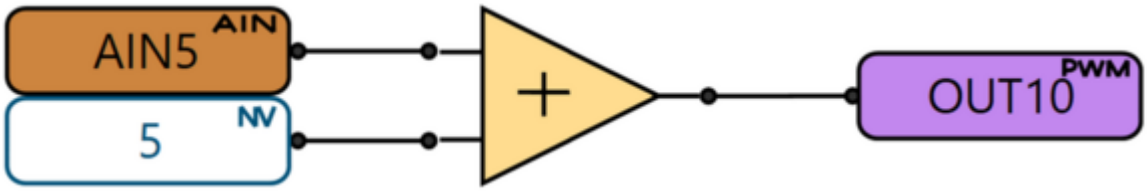

Описание

Компонент выполняет операцию сравнения численных величин или аналоговых сигналов. Если рассматривать пример подключения, то в том случае, когда уровень тока на выходе OUT1 равен 4-м амперам, на выходе OUT2 будет присутствовать высокий сигнал.

СЛОЖИТЬ



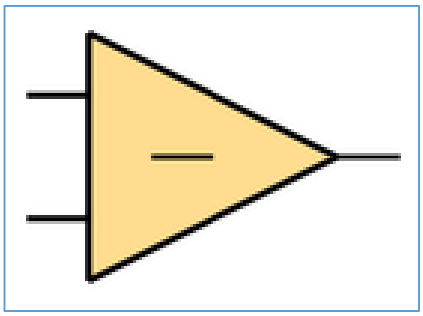
Пример подключения



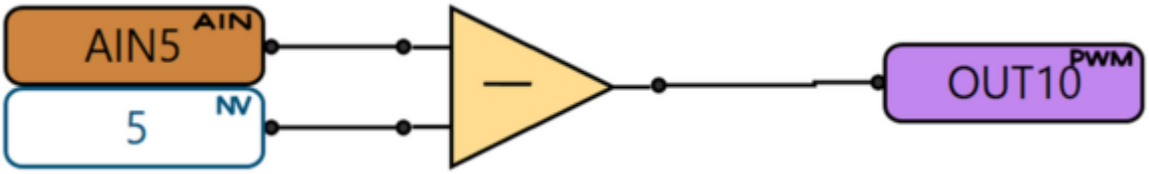

Описание

При подаче сигналов или установки численных значений на входах блока, на выходе блока будет выведен результат сложения. На примере подключения изображена схема, при которой на выходе ШИМ OUT10 будет присутствовать импульс, скважность которого определена как сумма значения аналогового сигнала, полученного со входа AIN5 и константы 5.

ВЫЧЕСТЬ



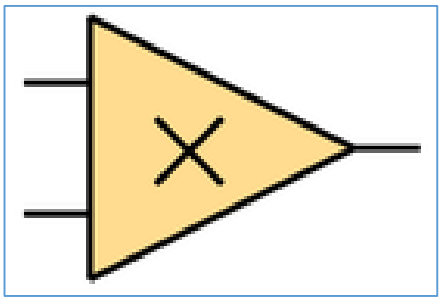
Пример подключения



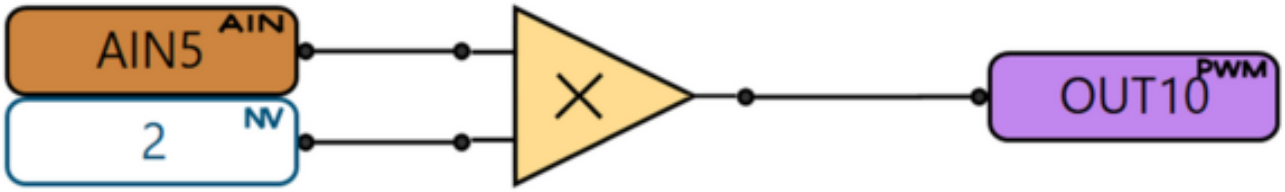

Описание

При подаче сигналов или установки численных значений на входах блока, на выходе блока будет выведен результат вычитания второго из первого. На примере подключения изображена схема, при которой на выходе ШИМ OUT10 будет присутствовать импульс, скважность которого определена как разность значения аналогового сигнала, полученного со входа AIN5 и константы 5.

УМНОЖИТЬ



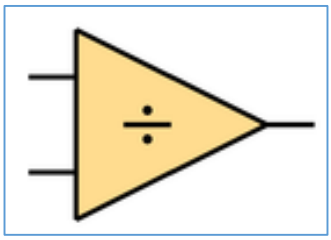
Пример подключения



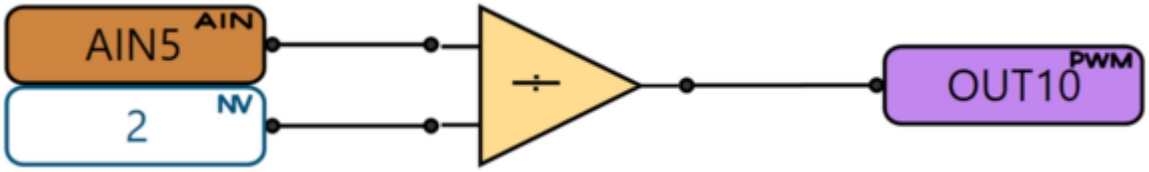

Описание

При подаче сигналов или установки численных значений на входах блока, на выходе блока будет выведен результат их перемножения. На примере подключения изображена схема, при которой на выходе ШИМ OUT10 будет присутствовать импульс, скважность которого определена как произведение значения аналогового сигнала, полученного со входа AIN5 и константы 2.

РАЗДЕЛИТЬ



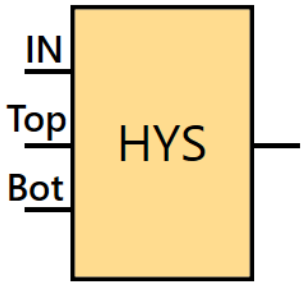
Пример подключения



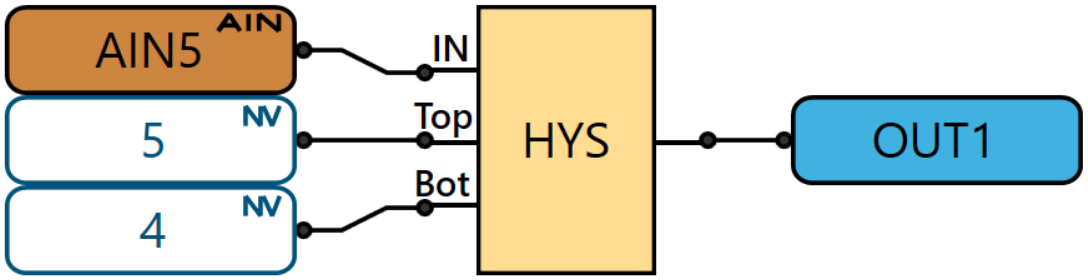

Описание

При подаче сигналов или установки численных значений, на выходе блока будет выведен результат деления значения с первого входа на второй. На примере подключения изображена схема, при которой на выходе ШИМ OUT10 будет присутствовать импульс, скважность которого определена как частное значения аналогового сигнала, полученного со входа AIN5 и константы 2.

ГИСТЕРЕЗИС



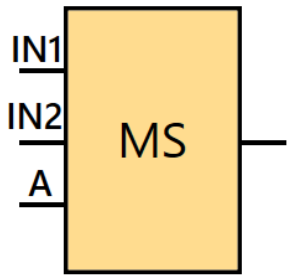

Пример подключения



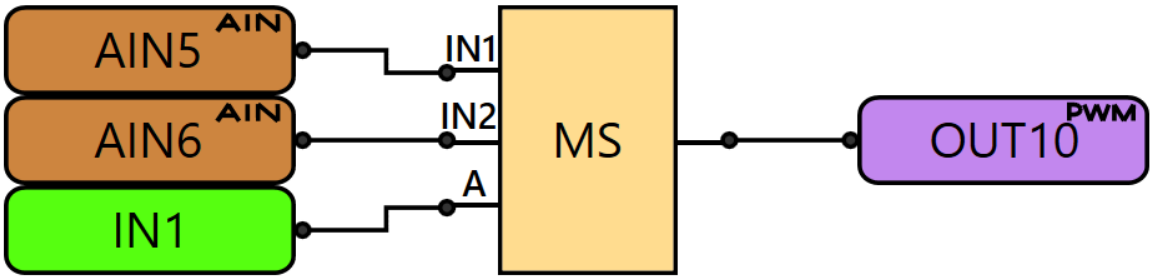

Описание

При подаче сигнала на вход IN, на выходе блока будет присутствовать высокий сигнал при превышении сигналом на входе IN значения на втором входе (Top). Высокий сигнал на выходе блока будет сохраняться до тех пор, пока значение сигнала IN не упадет до значения, указанного на третьем входе (Bot). На примере подключения изображена схема, при которой при наличии сигнала на AIN5 на выходе OUT1 будет высокий сигнал только если уровень сигнала на AIN5 превысил 5 В, сигнал на OUT1 сохраняется пока значение на AIN5 не опустится до 4.

МУЛЬТИПЛЕКСОР АНАЛОГОВЫЙ




Пример подключения

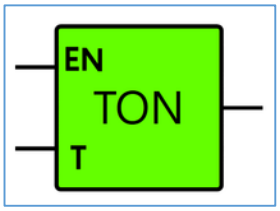



Описание

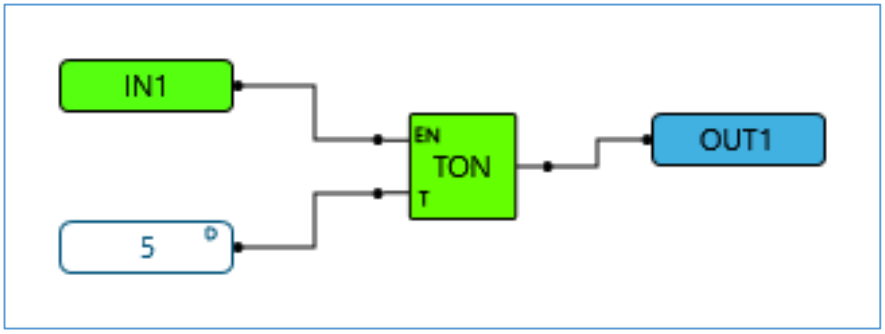
Компонент является простым двухвходовым мультиплексором, где A-управляющий вход. На примере подключения изображена схема, при которой на выходе ШИМ OUT10 будет присутствовать сигнал со скважностью, заданной как значение, полученное с аналогового входа AIN5 при низком значении на входе IN1 и заданной как значение, полученное с аналогового входа AIN6, при высоком значении на входе IN1.

## Временные

### ЗАДЕРЖКА НА ВКЛ




Пример подключения

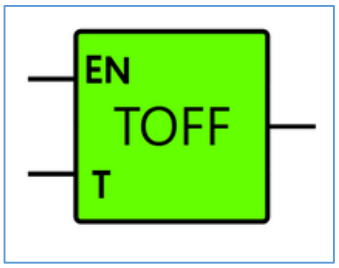



Описание

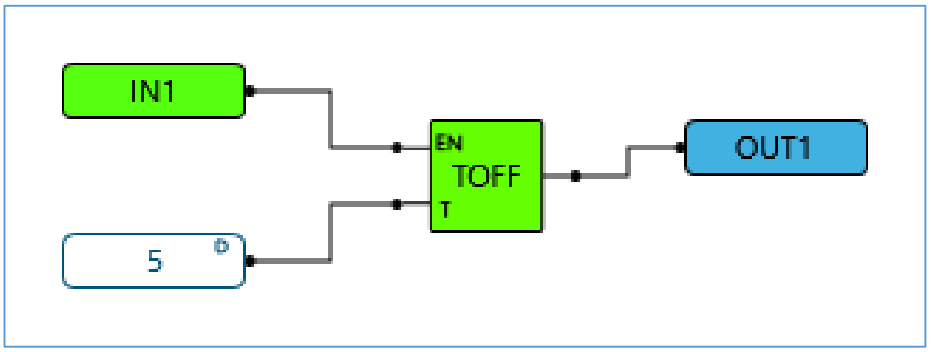
На примере подключения изображена схема, при которой после подачи фронта высокого сигнала на вход IN1, спустя задержку по времени в 5 секунд, на выходе OUT1 появляется высокий сигнал.

- Время устанавливается в секундах.
- С шагом 100 мс (0.1 с)
- Возможно использование точки в качестве разделителя. Например, "0.1"

ЗАДЕРЖКА НА ВЫКЛ




Пример подключения

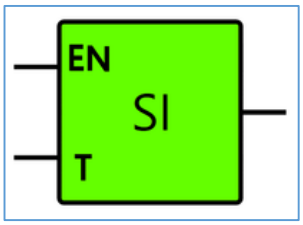



Описание

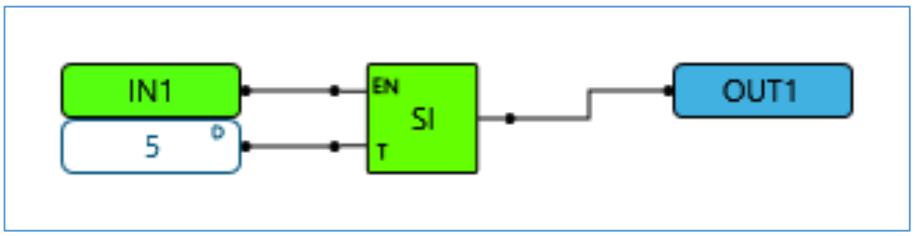
На примере подключения изображена схема, при которой после подачи фронта высокого сигнала на вход IN1, на выходе OUT1 появляется высокий сигнал, после чего, при подаче на вход IN1 низкого сигнала, на выходе OUT1 сохраняется высокий сигнал еще в течение 5 секунд.

- Время устанавливается в секундах.
- С шагом 100 мс (0,1 с)
- Возможно использование точки в качестве разделителя. Например, "0.1"

ЕДИНИЧНЫЙ ИМПУЛЬС




Пример подключения

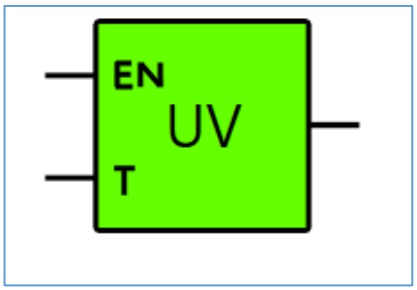



Описание

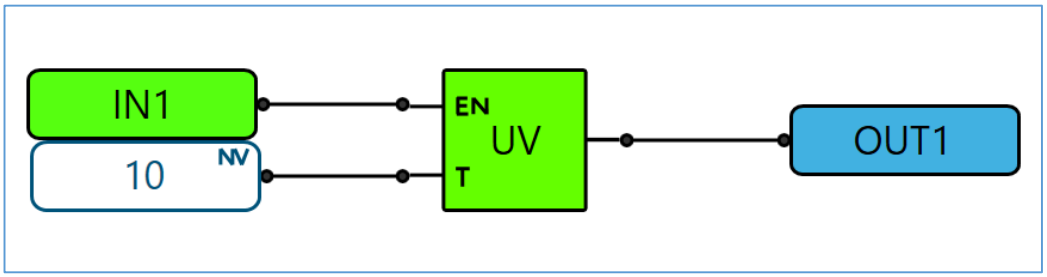
На примере подключения изображена схема, при которой, после подачи высокого сигнала на вход IN1, на выходе OUT1 устанавливается высокий сигнал в течение 5 секунд, при условии сохранения на IN1 высокого сигнала, далее-низкий. Отличие от блока ОДНОВИБРАТОР состоит в том, что при повторной подаче высокого сигнала на вход IN1 при уже установленном на OUT1 высокого состояния и начатом отсчете времени, отсчет не начинается заново.

- Время устанавливается в секундах.
- С шагом 100 мс (0,1 с)
- Возможно использование точки в качестве разделителя. Например, "0.1"

ОДНОВИБРАТОР




Пример подключения

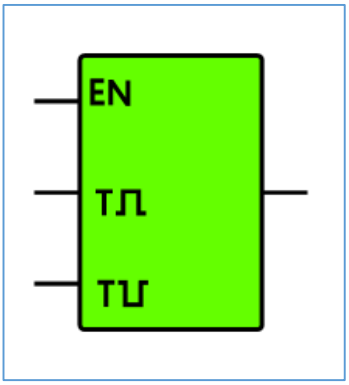



Описание

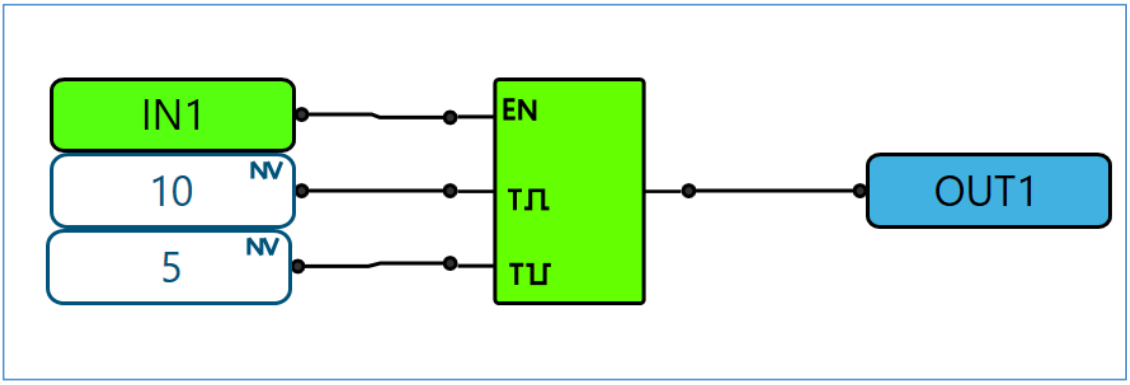
На примере подключения изображена схема, при которой после подачи высокого сигнала на вход IN1, на выходе OUT1 устанавливается высокий сигнал в течение 5 секунд, далее-низкий. Отличие от блока ЕДИНИЧНЫЙ ИМПУЛЬС состоит в том, что при повторной подаче высокого сигнала на вход IN1 при уже установленном на OUT1 высоком состоянии и начатом отсчете времени, отсчет сбрасывается и начинается заново.

- Время устанавливается в секундах.
- С шагом 100 мс (0,1 с)
- Возможно использование точки в качестве разделителя. Например, "0.1"

ИМПУЛЬС




Пример подключения



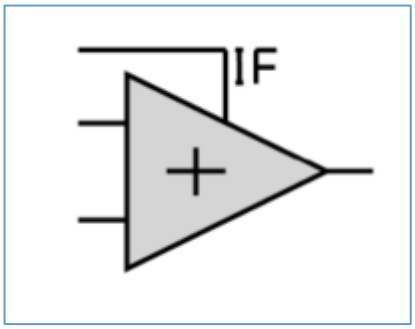

Описание

На примере подключения изображена схема, при которой после подачи высокого сигнала на вход IN1, на выходе OUT1 будет присутствовать высокий сигнал в течение 5 секунд, после чего в течении 10 секунд будет присутствовать низкий сигнал, далее снова высокий в течении 5 секунд и так далее.

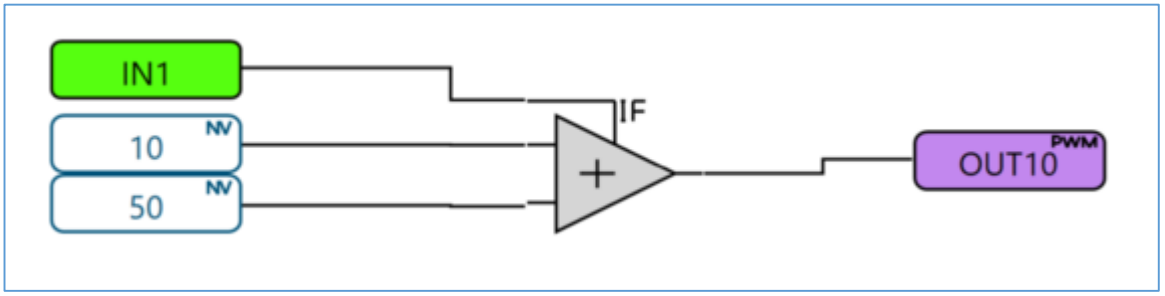
- Время устанавливается в секундах.
- С шагом 100 мс (0,1 с)
- Возможно использование точки в качестве разделителя. Например, "0.1"

## Специальные

### СЛОЖИТЬ ЕСЛИ



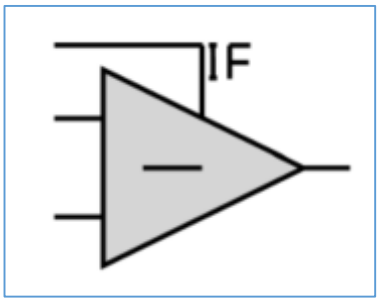

Пример подключения



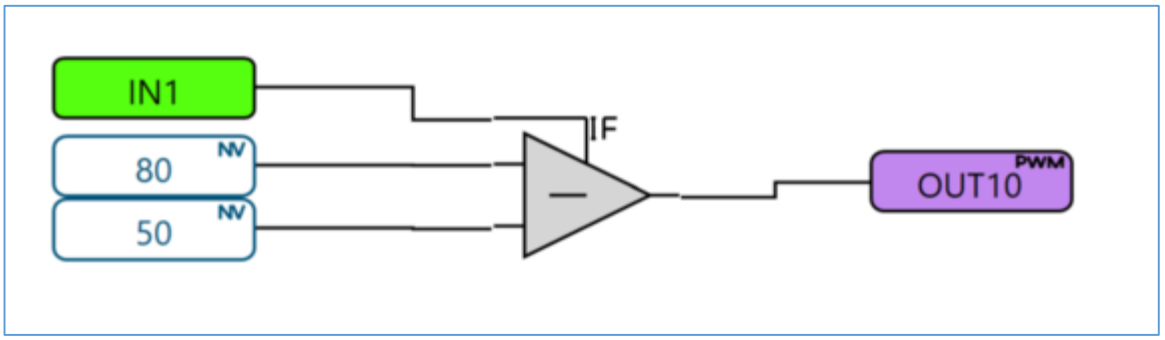

Описание

При условии подачи высокого сигнала на вход IF, сигналы, поступающие на входы 1 и 2 складываются. На примере подключения изображена схема, при которой при подаче высокого сигнала на вход IN1, на выходе ШИМ OUT10 будет присутствовать импульсный сигнал с коэффициентом заполнения 60%.

ВЫЧЕСТЬ ЕСЛИ



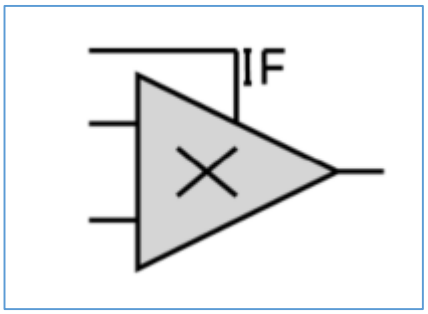

Пример подключения



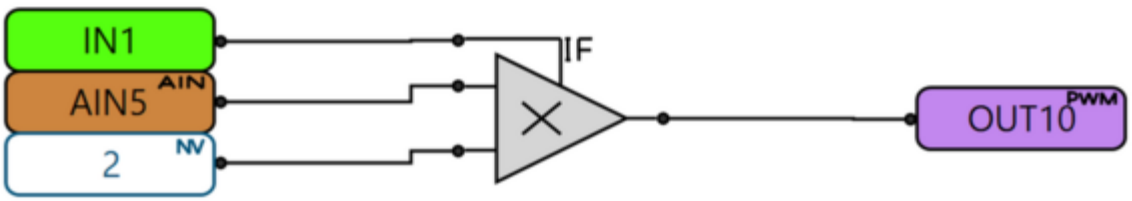

Описание

При условии подачи высокого сигнала на вход IF, сигналы, поступающие на входы 1 и 2 вычитаются. На примере подключения изображена схема, при которой при подаче высокого сигнала на вход IN1, на выходе ШИМ OUT10 будет присутствовать импульсный сигнал с коэффициентом заполнения 30%.

УМНОЖИТЬ ЕСЛИ



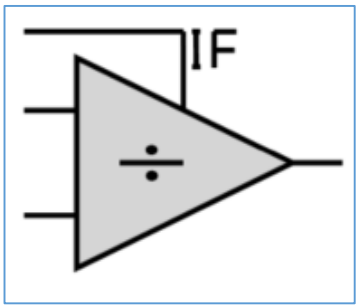

Пример подключения



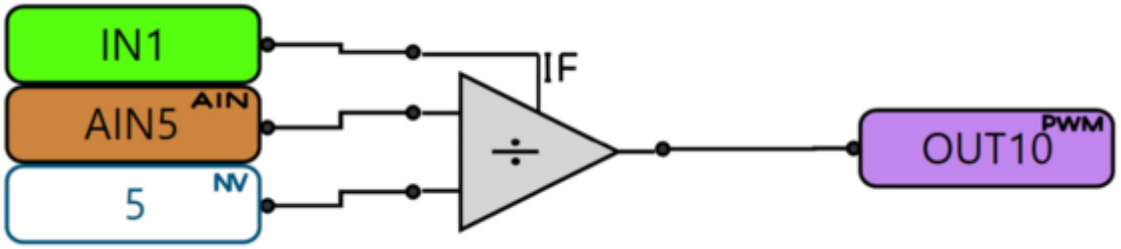

Описание

При условии подачи высокого сигнала на вход IF, сигналы, поступающие на входы 1 и 2 перемножаются. На примере подключения изображена схема, при которой при подаче высокого сигнала на вход IN1, на выходе ШИМ OUT10 будет присутствовать импульсный сигнал с коэффициентом заполнения равным произведению сигнала с аналогового входа AIN5 и константы 2.

РАЗДЕЛИТЬ ЕСЛИ



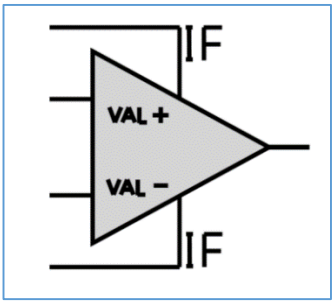

Пример подключения



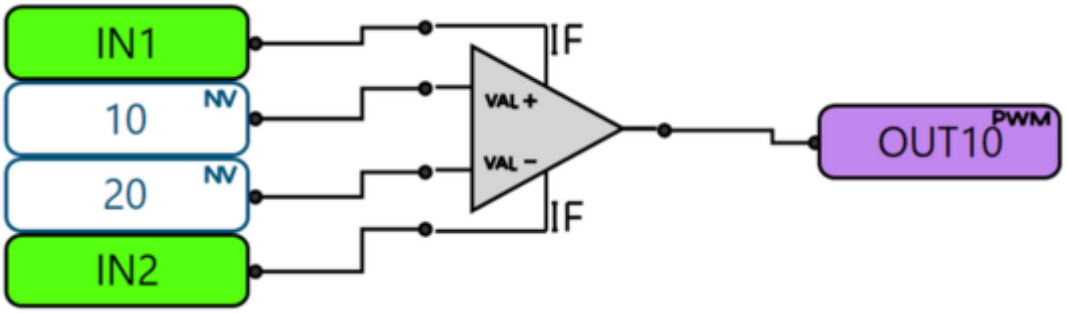

Описание

При условии подачи высокого сигнала на вход IF, сигналы, поступающие на входы 1 и 2 делятся первый на второй. На примере подключения изображена схема, при которой при подаче высокого сигнала на вход IN1, на выходе ШИМ OUT10 будет присутствовать импульсный сигнал с коэффициентом заполнения равным частному от деления сигнала с аналогового входа AIN5 на 5.

СЛОЖИТЬ/ВЫЧЕСТЬ ЕСЛИ



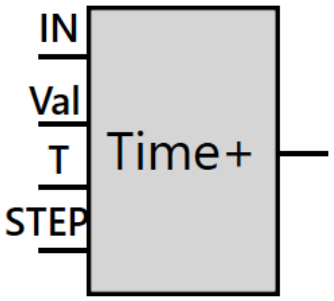

Пример подключения



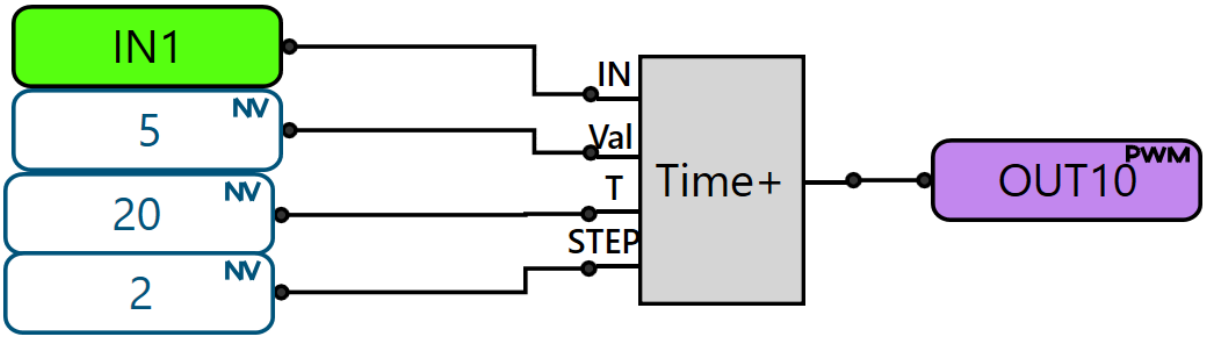

Описание

При фронте импульса на верхнем выводе IF, к величине на выходе блока прибавится значение, заданное на первом входе. При фронте импульса на нижнем выводе IF, от величины на выходе блока отнимется значение, заданное на втором входе. На примере подключения изображена схема, при которой при подаче высокого сигнала на IN1, коэффициент заполнения ШИМ на OUT10 возрастет на 10%, а при подаче высокого сигнала на IN2, коэффициент заполнения ШИМ на OUT10 уменьшится на 20%.

СЛОЖИТЬ ПО ВРЕМЕНИ



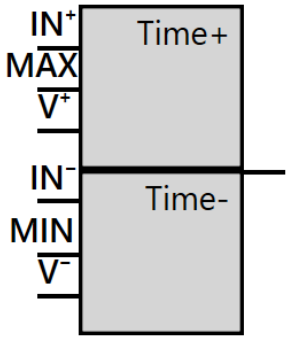

Пример подключения



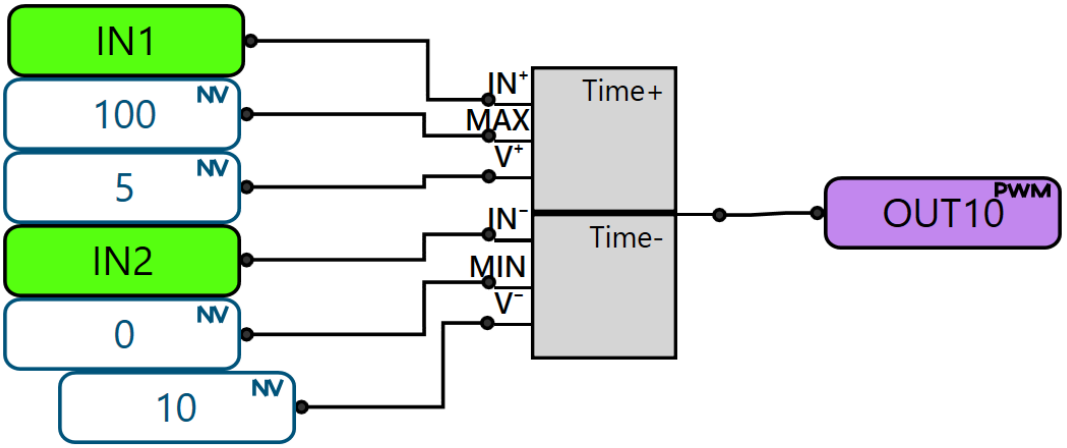

Описание

Компонент выполняет операцию прибавления указанной на входе VAL величины к выходному значению с заданным на входе STEP временны шагом до тех пор, пока значение времени не достигнет значения, казанного на входе T. На примере подключения изображена схема, при которой при подаче сигнала на IN1, коэффициент заполнения ШИМ на выходе OUT10 будет увеличиваться на 5% каждые 2 секунды пока не пройдет 20 секунд. Таким образом алгоритм пройдет 10 операций сложения и спустя 20 секунд на выходе OUT10 будет присутствовать импульс с коэффициентом заполнения 50%.

СКОРОСТЬ РОСТА/СПАДА (увеличение и уменьшение значения с определенной скоростью)



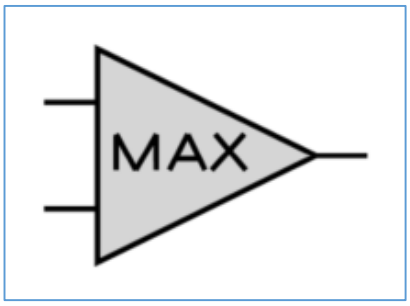

Пример подключения



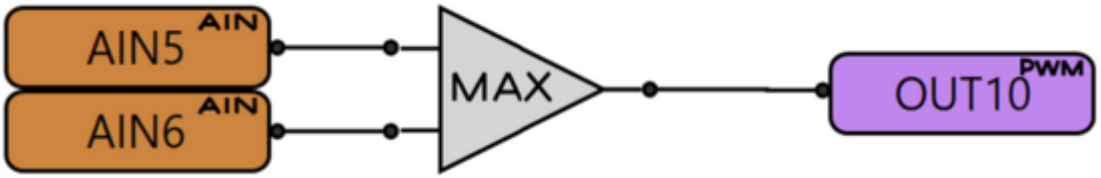

Описание

Компонент выполняет операцию задания скорости роста/спада величины импульса. На примере подключения изображена схема, при которой при подаче высокого сигнала на вход IN1 на выходе ШИМ OUT10 будет образовываться сигнал с коэффициентом заполнения, увеличивающимся на 5% за один цикл выполнения программы, пока не достигнет 100%. Цикл выполнения задается при выборе значений таймера функции. При подаче высокого сигнала на вход IN2 на выходе ШИМ OUT10 будет образовываться сигнал с коэффициентом заполнения, уменьшающимся на 10% за один цикл выполнения программы пока не достигнет 0%.

МАКС



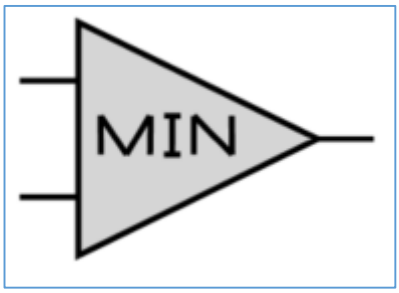

Пример подключения



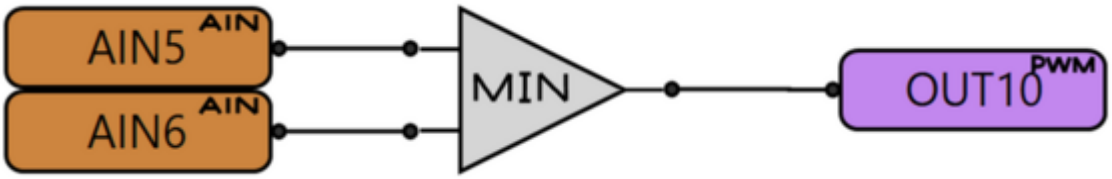

Описание

Компонент выдает на выходе максимальное из двух входных значений. На примере подключения изображена схема, при которой производится сравнение величин на аналоговых входах AIN5 и AIN6 и установка коэффициента заполнения ШИМ на выходе OUT10 равного максимальному из двух значений.

МИН



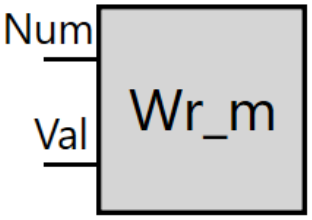

Пример подключения



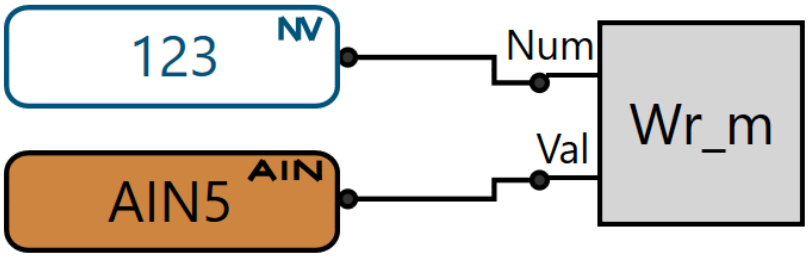

Описание

Компонент выдает на выходе минимальное из двух входных значений. На примере подключения изображена схема, при которой производится сравнение величин на аналоговых входах AIN5 и AIN6 и установка коэффициента заполнения ШИМ на выходе OUT10 равного минимальному из двух значений.

ЗАПИСЬ В ПАМЯТЬ



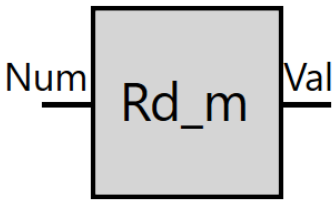

Пример подключения




Описание

Компонент осуществляет запись значения V по адресу Num в EEPROM.

ЧТЕНИЕ ИЗ ПАМЯТИ

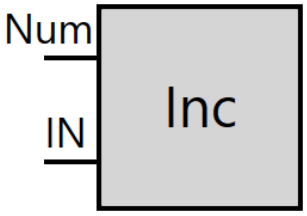



Пример подключения

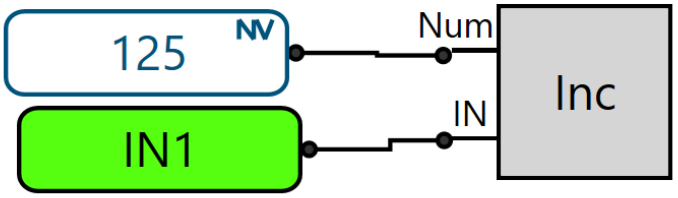

Описание

Компонент осуществляет чтение значения по адресу Num и передачу его на подключенный выход.

ИНКРЕМЕНТ



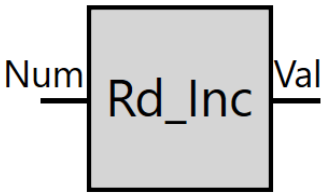

Пример подключения



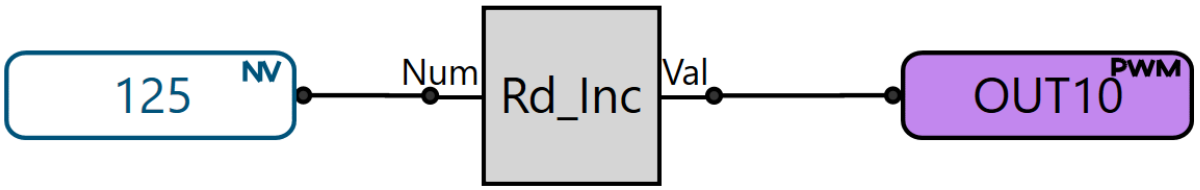

Описание

Компонент осуществляет инкремент (увеличение на 1) значения по адресу Num.

ЗНАЧЕНИЕ ИНКРЕМЕНТА



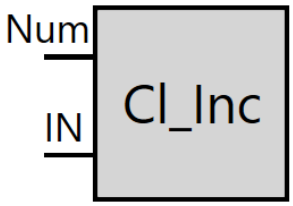

Пример подключения



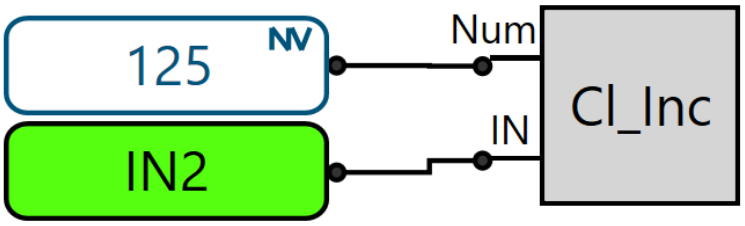

Описание

Компонент осуществляет чтение значения инкремента по адресу Num и передача его на выход Val.

СБРОС ИНКРЕМЕНТА




Пример подключения




Описание

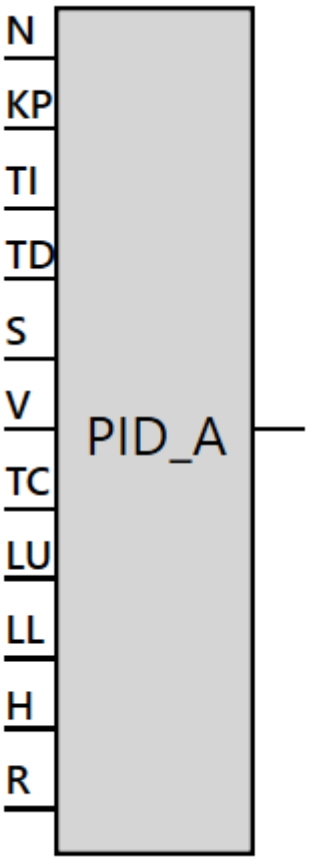
Компонент осуществляет сброс ранее записанного по адресу Num инкремента при подаче сигнала на вход IN.

Пропорционально-интегрально-дифференцирующие (ПИД) регуляторы

ПИД-регуляторы предназначены для поддержания с требуемой точностью состояния объекта управления путём выдачи управляющего воздействия, вычисленного на основании сравнения величины регулируемого параметра объекта с заданным значением и свойствами объекта (инерционностью, линейностью). Расчёт коэффициентов регулятора выполняется в соответствии с общепринятыми методиками настройки.

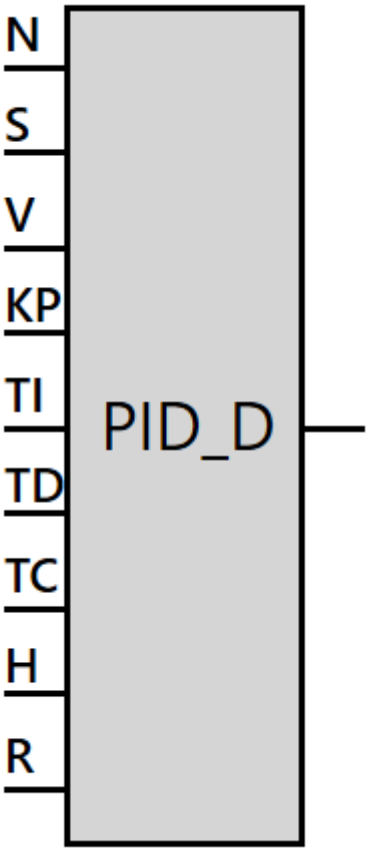
На данный момент вы можете использовать до 6 регуляторов одного типа в проекте, номер устанавливается как блок «ЧИСЛО» на входе N.

Блок PID_A (аналоговый выход)



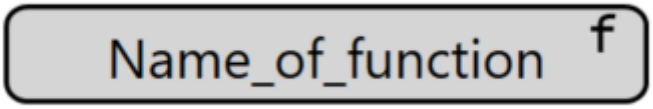

- N (номер регулятора, 1-6)
- KP (пропорциональный коэффициент, 0.01…100)
- TI (интегральная постоянная, KP*период работы)
- TD (дифференциальная постоянная, KP/период работы)
- S (уставка, единиц, 0…32767)
- V (измеряемая величина, 0…32767)
- TC (период работы, секунд, с дискретностью цикла АСУ (0.02-0.1с), 0 – 30000с)
- LU (ограничение выхода сверху, например 100 для ШИМ-выхода)
- LL (ограничение выхода снизу, например 0 для ШИМ-выхода)
- H (зона нечувствительности, гистерезис)
- R (сброс, выход устанавливается в значение LL)

Блок PID_D (аналоговый выход)

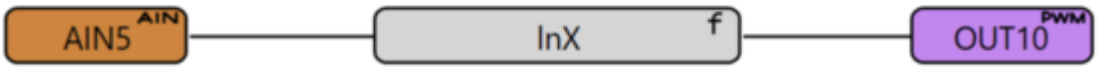



- N (номер регулятора, 1-6)
- KP (пропорциональный коэффициент, 0.01…100)
- TI (интегральная постоянная, KP*период работы)
- TD (дифференциальная постоянная, KP/период работы)
- S (уставка, единиц, 0…32767)
- V (измеряемая величина, 0…32767)
- TC (период работы, секунд, с дискретностью цикла АСУ (0.02-0.1с), 0 – 30000с)
- H (зона нечувствительности, гистерезис)
- R (сброс, выход устанавливается в значение 0)

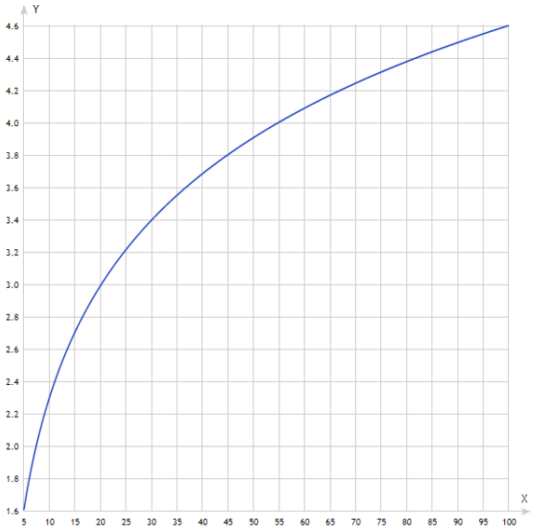
Пользовательская функция (алгоритм задается пользователем)




Пример подключения



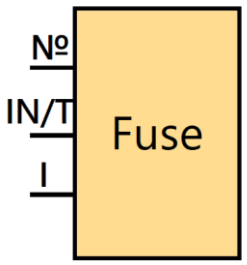

Выходная диаграмма



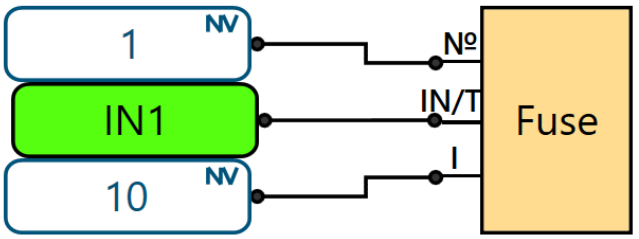

Описание

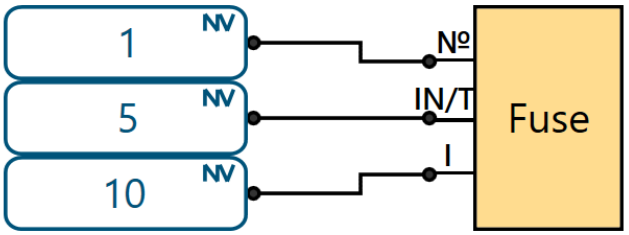
Функция, алгоритм работы которой задается пользователем. В примере показана пользовательская функция, берущая натуральный алгоритм от значения, полученного с аналогового входа и выдающая результат на выход ШИМ. Перечень возможных используемых функций в теле функции (например sin, log, tan и т.д.) соответствует библиотеке math языка СИ. Принцип создания пользовательских функций будет описан в соответствующем разделе.

МГНОВЕННЫЙ — возврат в исходное состояние при достижении IN/T




Пример подключения

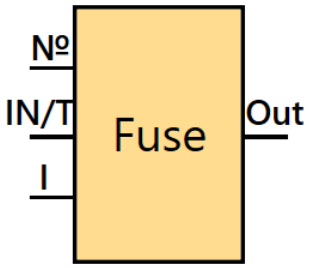


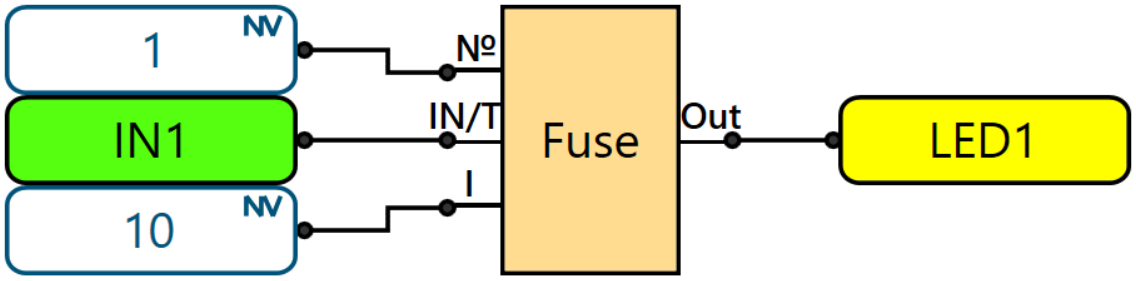

Описание

Мгновенный предохранитель, защищающий указанный под номером на первом входе выход от скачков тока. Ко второму входу блока подключается цифровой вход, по которому осуществляется восстановление предохранителя или задается время по прошествии которого осуществляется восстановление. На примере изображена схема, при которой при превышении тока на выходе под номером 1 (OUT1) свыше 10 ампер сработает блокировка и выход будет отключен от цепи до тех пор, пока на вход IN1 не будет подан высокий сигнал. На втором примере при превышении тока свыше 10 ампер сработает блокировка и выход будет отключен от цепи в течение 5 секунд.

МГНОВЕННЫЙ+LED — мгновенный предохранитель с выходом



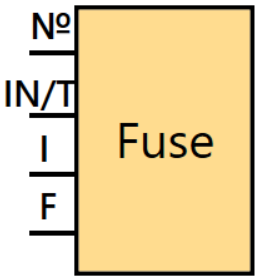

Пример подключения



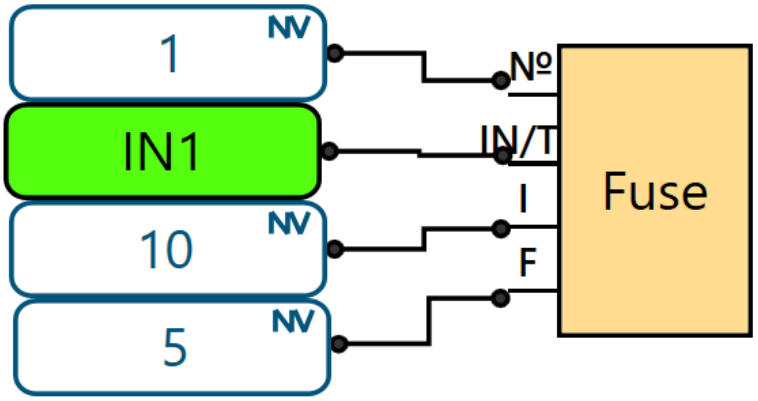

Описание

Мгновенный предохранитель с возможностью подключения выхода для определения срабатывания. На примере подключения изображена схема, при которой при превышении тока на выходе под номером 1 (OUT1) свыше 10 ампер сработает блокировка, на модуле загорится светодиод LED1 и выход будет отключен от цепи до тех пор, пока на вход IN1 не будет подан высокий сигнал.

ИНЕРЦИОННЫЙ— возврат в исходное состояние при достижении параметра IN/T



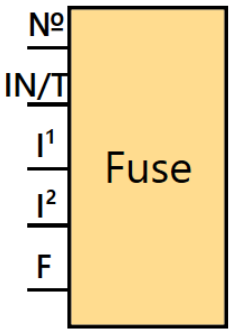

Пример подключения



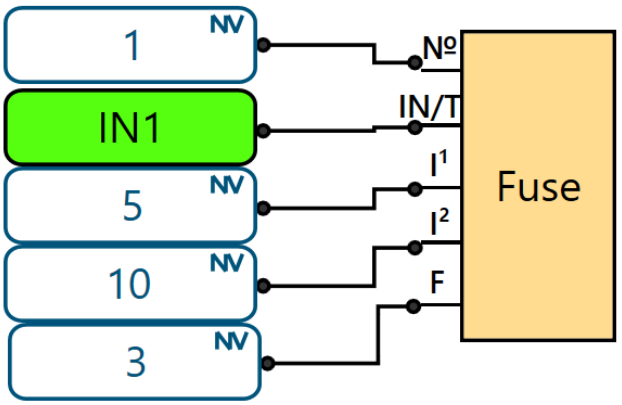

Описание

Инерционный предохранитель, защищающий указанный под номером на первом входе выход от скачков тока. Ко второму входу блока подключается цифровой вход, по которому осуществляется восстановление предохранителя или задается время по прошествии которого осуществляется восстановление. По третьему входу задается величина тока срабатывания. Вход Factor определяет период срабатывания функции.

ГИБРИДНЫЙ – сочетание мгновенного и инерционного предохранителей




Пример подключения



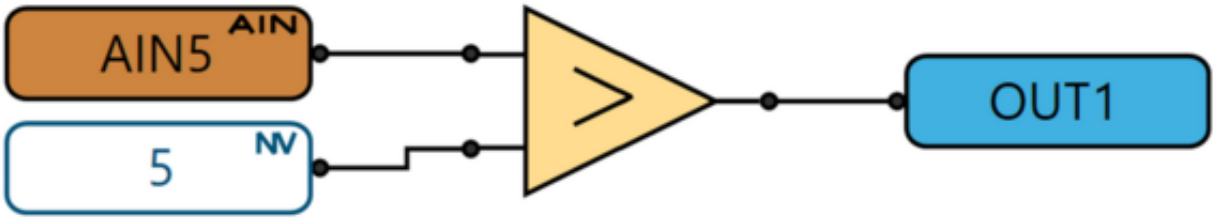

Описание

Блок ГИБРИДНЫЙ предохранитель является совокупностью блоков МГНОВЕННЫЙ и ИНЕРЦИОННЫЙ с возможностью указания токов для мгновенного (I1) и инерционного (I2) срабатывания.

## Постоянные величины

ЧИСЛО-установка числового значения


Пример подключения

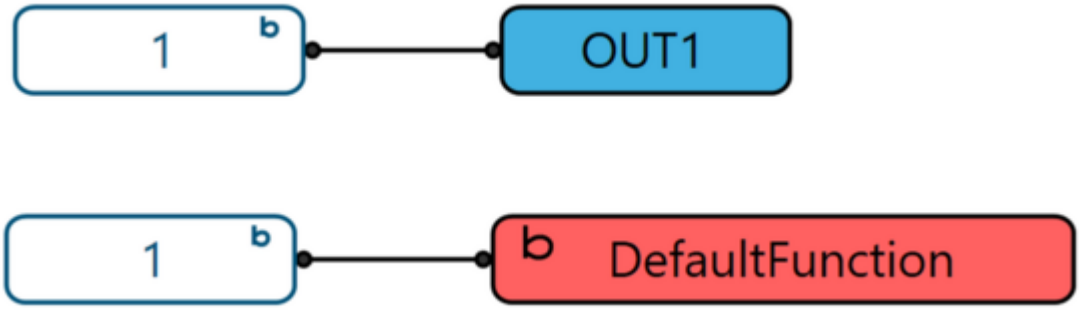



Описание

Блок ЧИСЛО позволяет задать постоянную числовую величину. На примере подключения изображена схема сравнения величины измеренного сигнала с аналогового входа AIN5 с заданной числовой величиной 5. Число может иметь целую часть и десятичные доли. Десятичная часть числа может отделяться как точкой, так и запятой.

БИТ

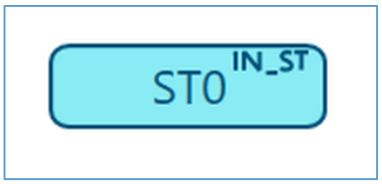

Пример подключения




Описание

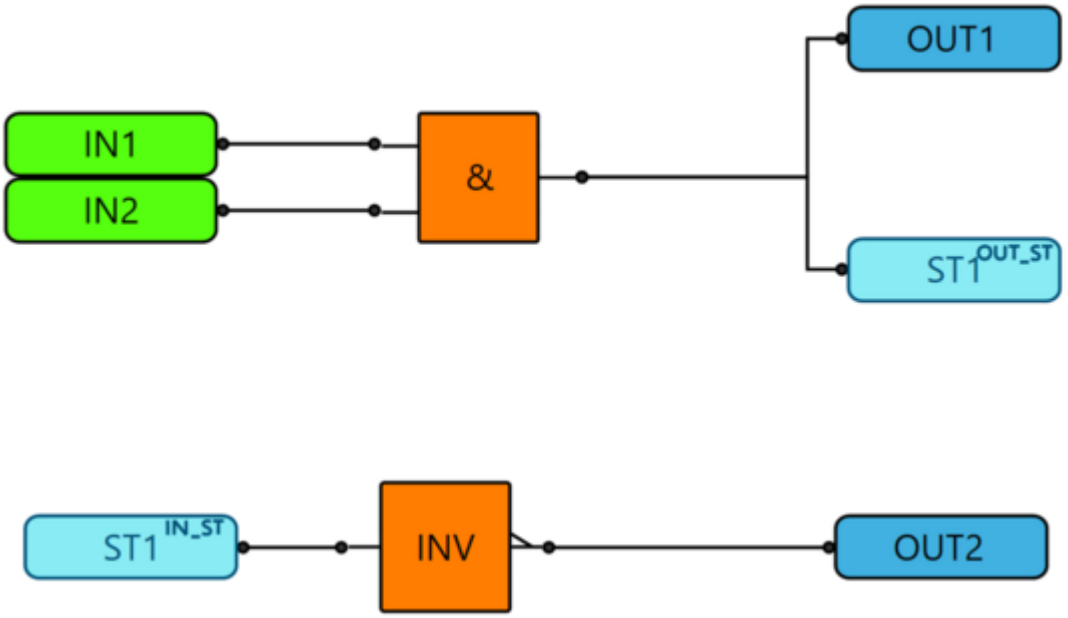
Позволяет задать логический 0 или 1. На примере подключения изображена схема, при которой на выходе OUT1 всегда присутствует высокий сигнал, а сообщение DefaultFunction отправляется согласно периоду выполнения функции.

ПОВТОРИТЕЛЬ ВХОДА / ПОВТОРИТЕЛЬ ВЫХОДА





Пример подключения



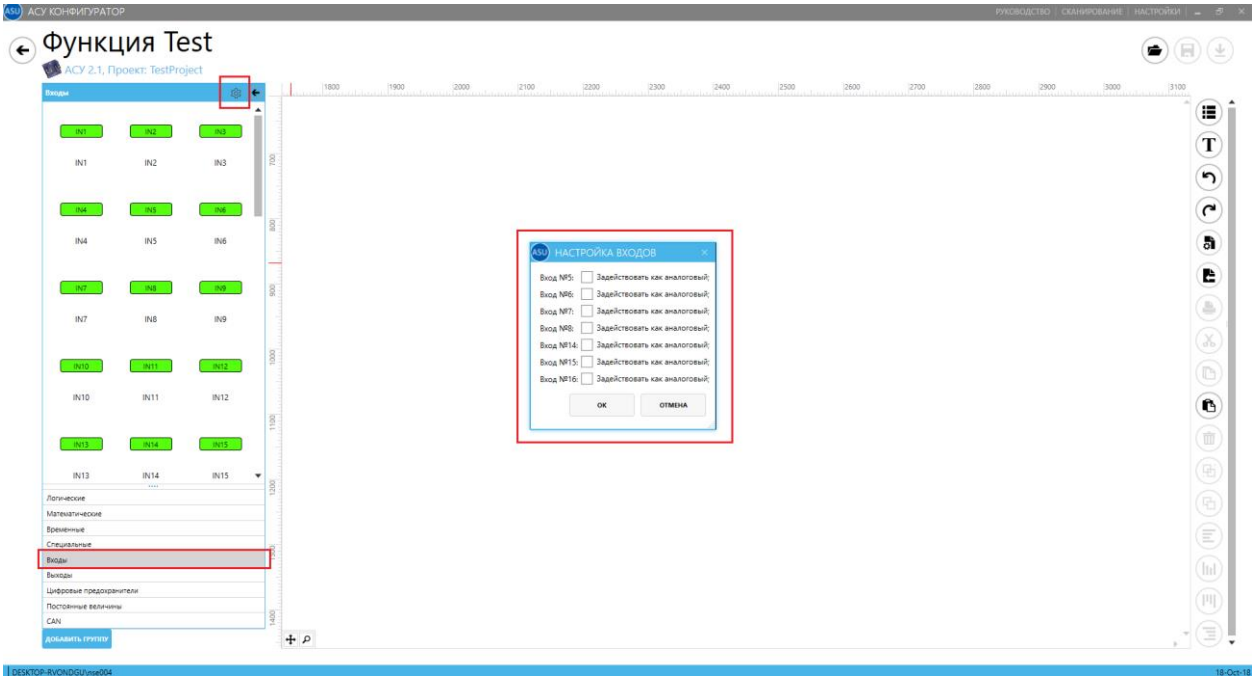

Описание

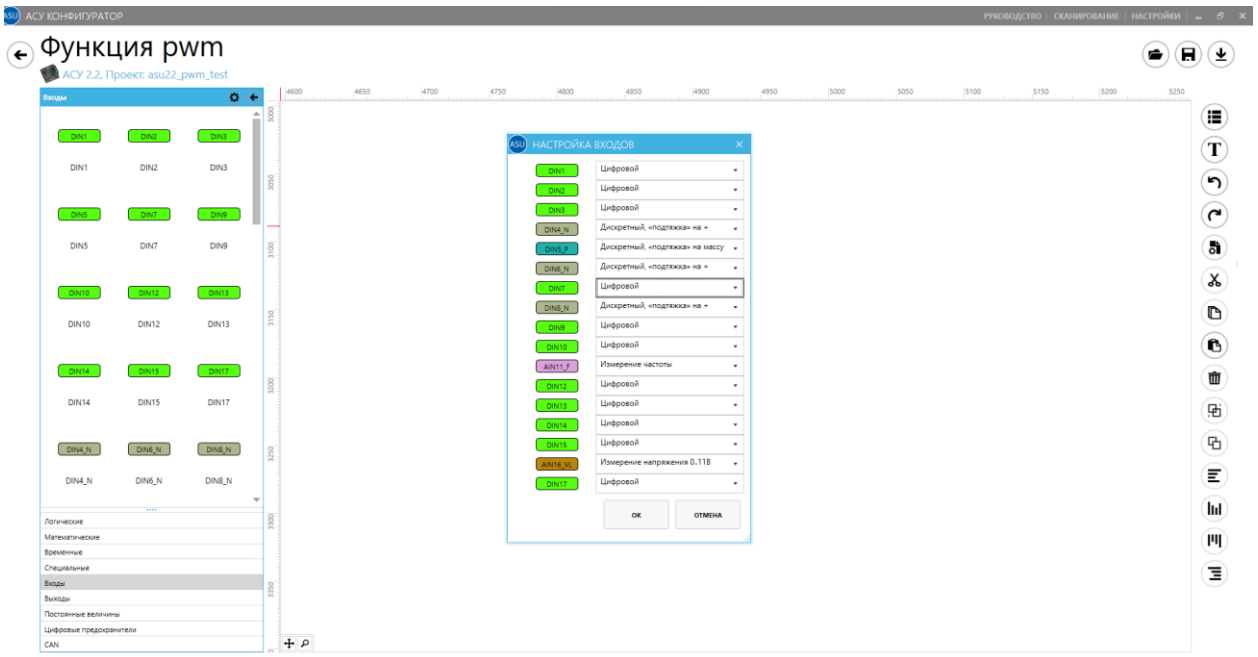
ПОВТОРИТЕЛЬ ВХОДА и ПОВТОРИТЕЛЬ ВЫХОДА используются для разноски сигналов с целью упрощения схемы. На примере подключения изображена схема, при которой сигнал с компонента «И» приходит на выход OUT1 и ПОВТОРИТЕЛЬ ВЫХОДА ST1. ПОВТОРИТЕЛЬ ВХОДА ST1 дублирует сигнал с компонента «И» и подает его на компонент «НЕ», который инвертирует его для выхода OUT2. Таким образом, при подаче высокого сигнала на входы IN1 и IN2, на выходе OUT1 будет высокий сигнал, а на выходе OUT2-низкий. Для определения принадлежности повторителей входа и выхода, необходимо задать их имена, произведя двойной клик мыши на компоненте.

# Настройка входов и выходов

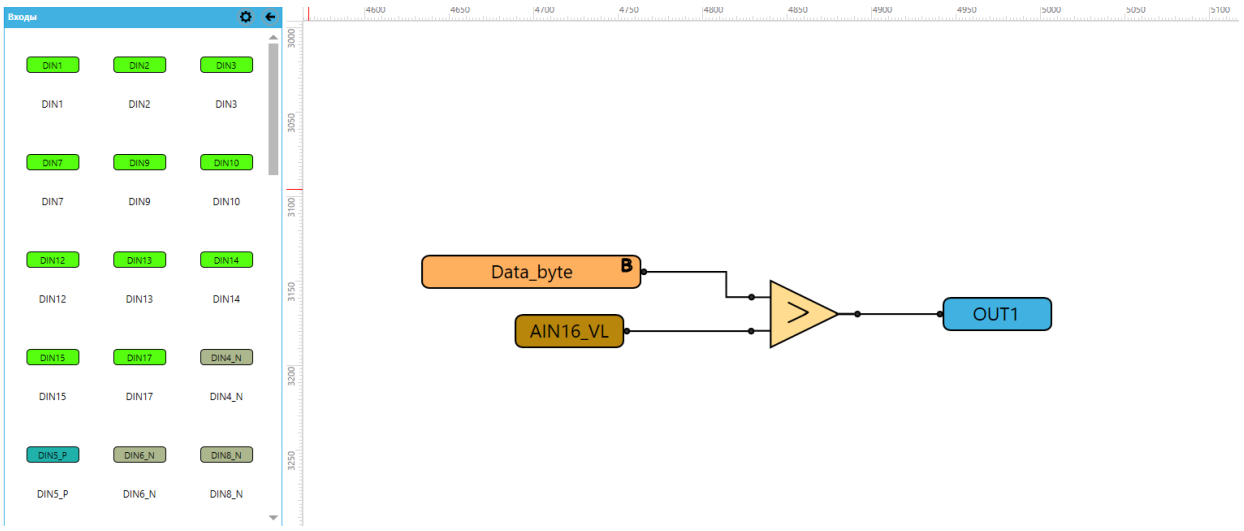
В некоторых модулях, таких как АСУ 2.1 и АСУ 2.2 доступно переключение цифрового входа на аналоговый и цифрового выхода на ШИМ-выход. В случае с АСУ 2.2 количество вариаций входов значительно увеличилось, подробнее о них вы можете узнать в документации к модулю.

Для переключения входа на аналоговый необходимо перейти во вкладку настроек группы «Входы». Откроется диалоговое окно с возможностью переключения доступных входов.



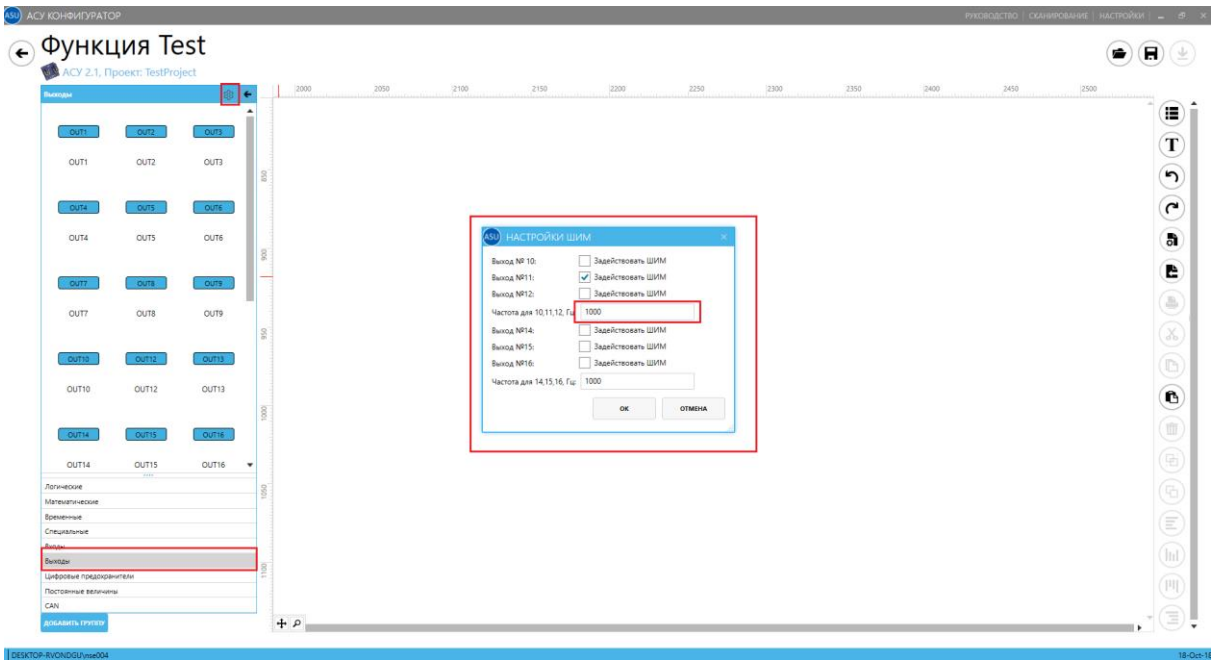



Пример использования аналогового входа.

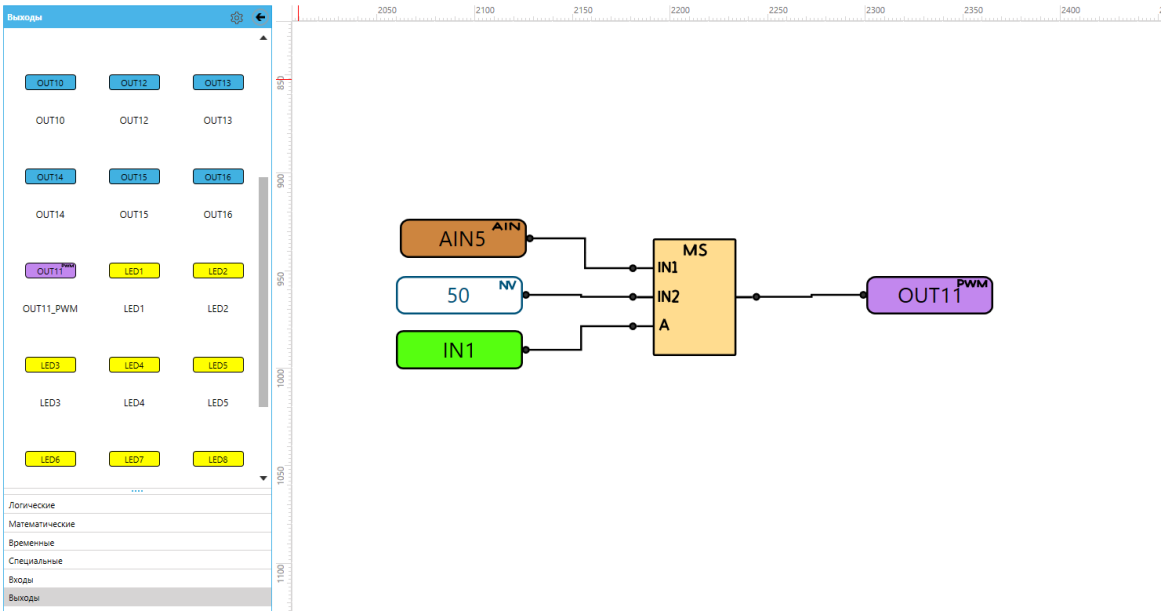



При данном алгоритме работы, на выходе OUT1 будет присутствовать высокий сигнал в том случае, если значение, пришедшее в байте CAN-сообщения Data_byte больше значения на аналоговом входе AIN16_VL.

Для переключения выхода в режим ШИМ необходимо выполнить аналогичные действия, а также установить частоту работы выхода.



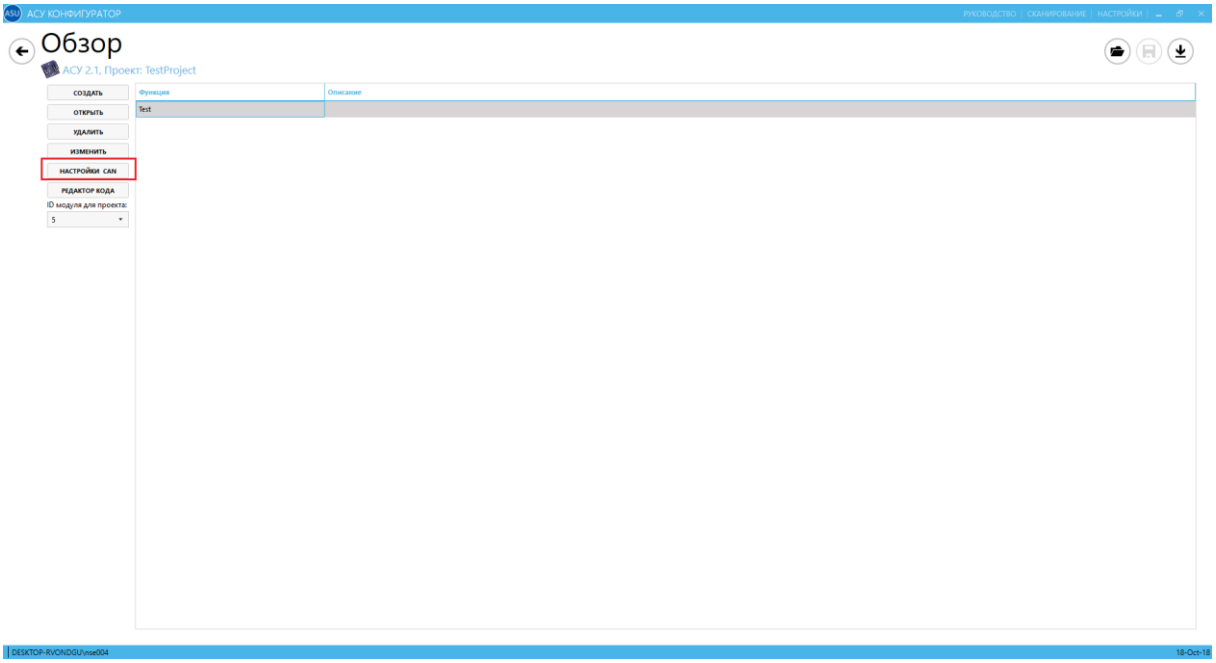

Пример использования аналогового входа и ШИМ-выхода

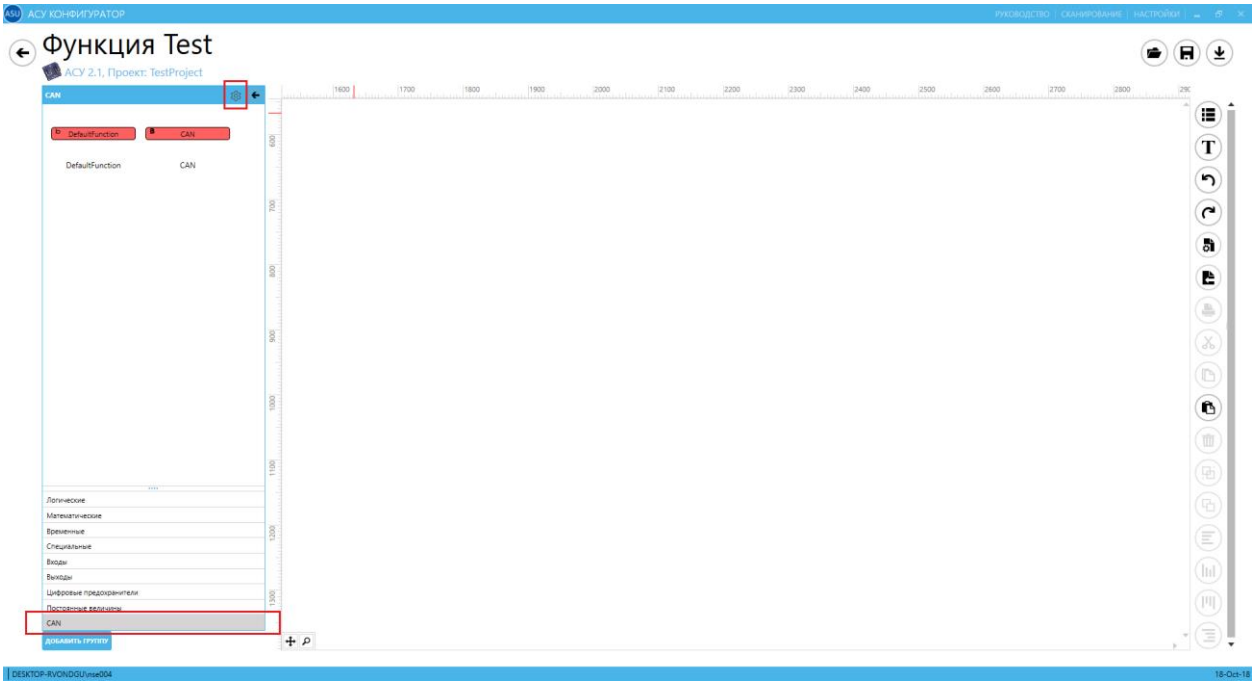



При данном алгоритме работы, на выходе OUT11 будет присутствовать ШИМ-сигнал с коэффициентом заполнения, равным значению, переданному на AIN5, при наличии на IN1 высокого сигнала. При низком сигнале IN1, на выходе будет присутствовать ШИМ-сигнал с коэффициентом заполнения=50%.

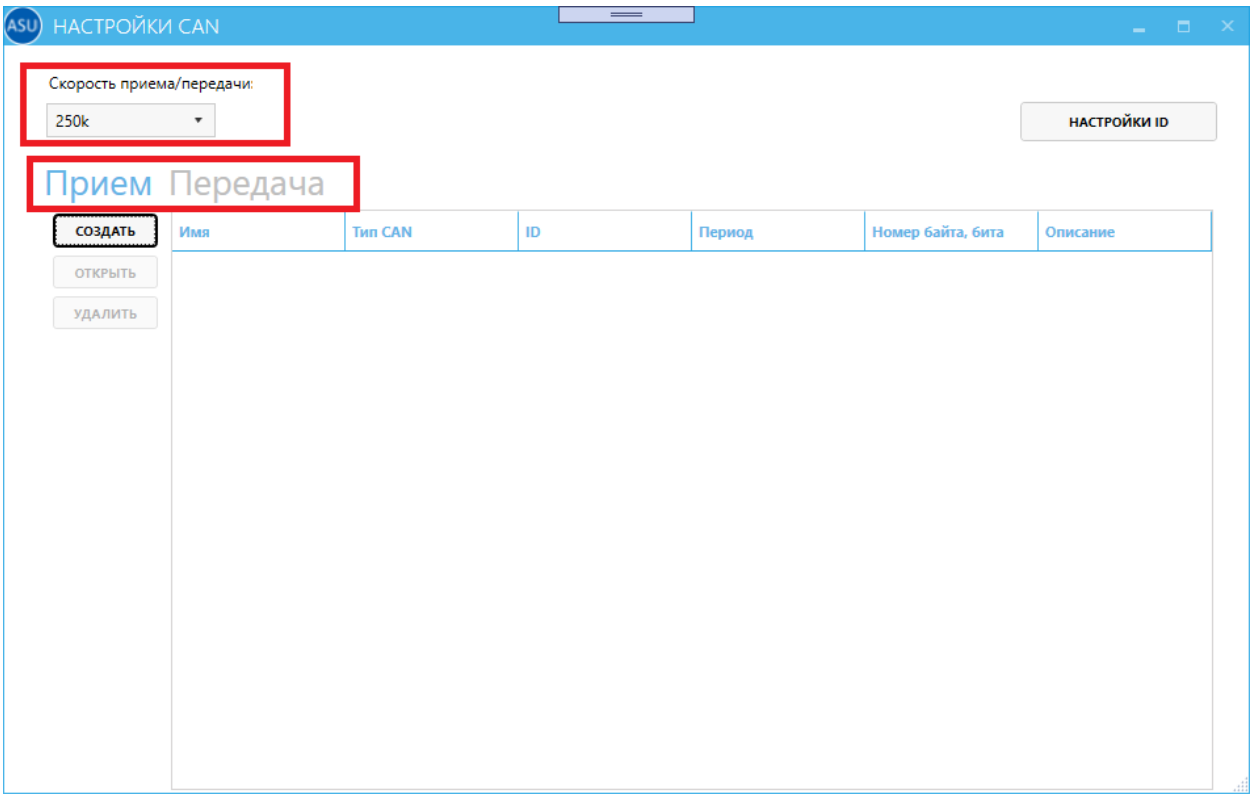
# Создание CAN-функций

Для создания CAN-функции, необходимо перейти в настройки CAN, сделать это можно из окна функций или зайдя в настройки группы «CAN».

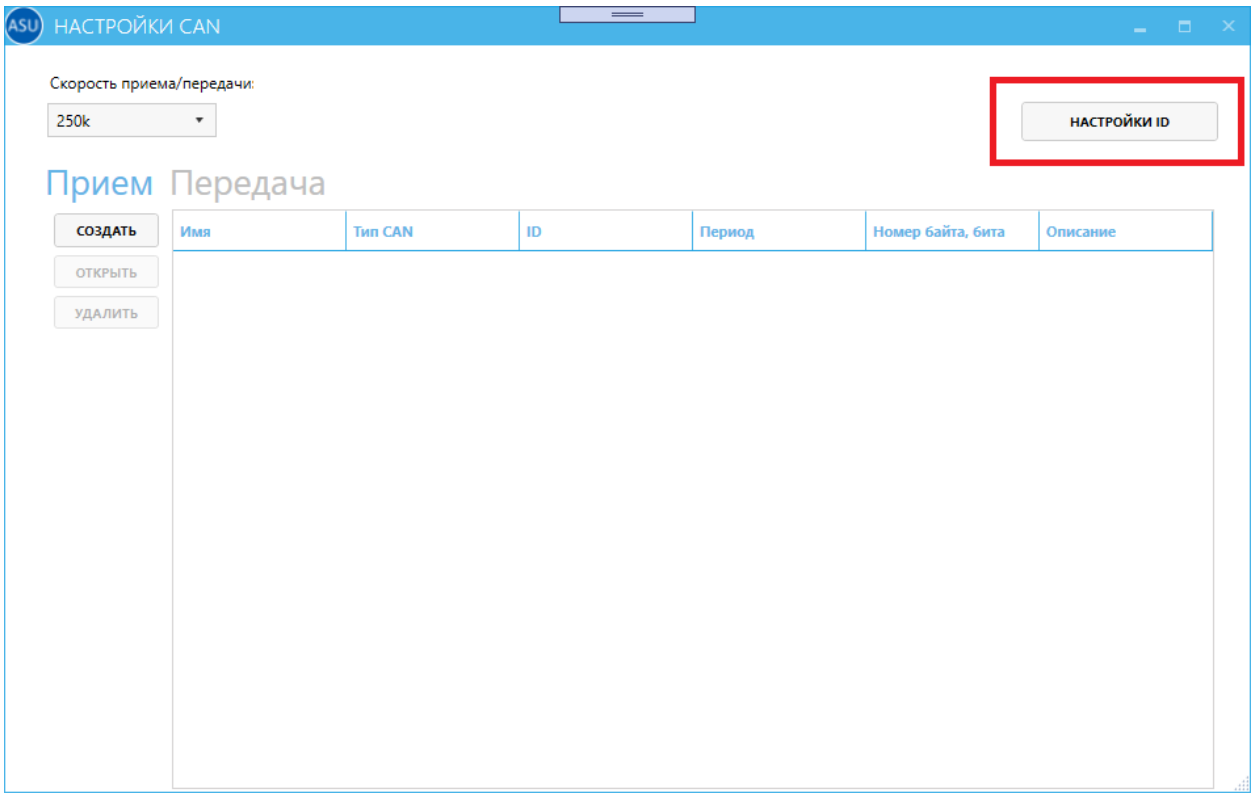


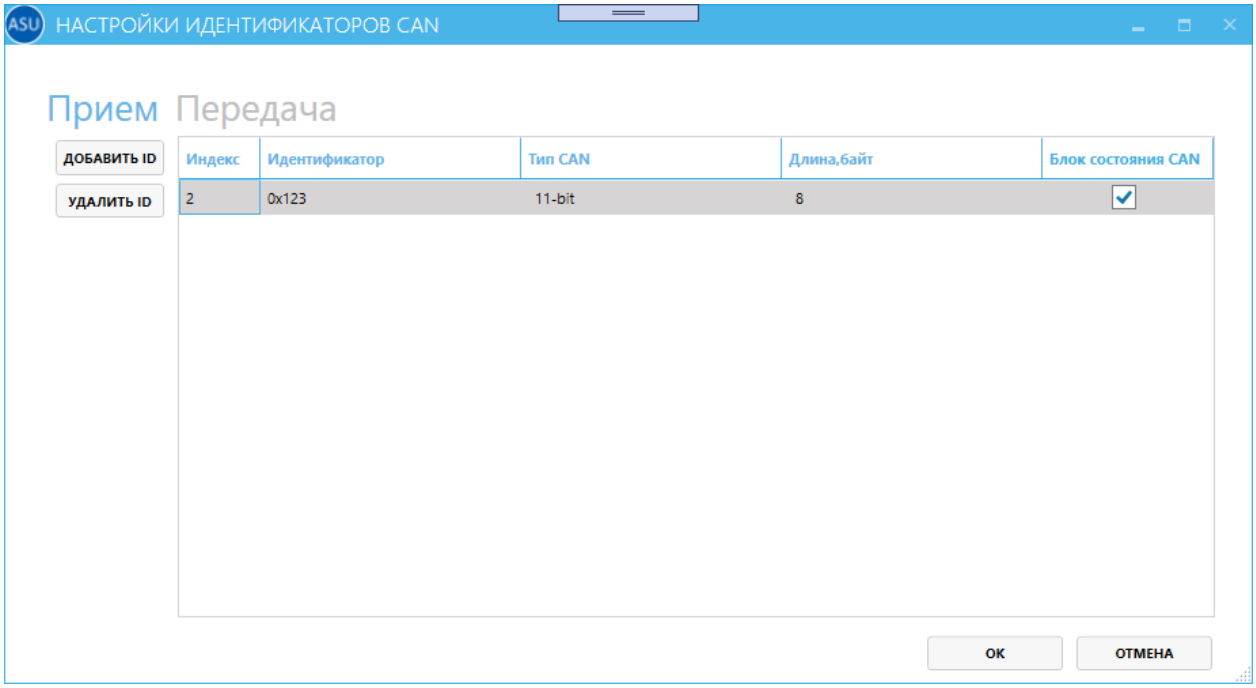
В настройках доступно создание функций приема и передачи, а также настройка скорости работы модуля, задаваемой при прошивке. Например, если модуль работал на скорости 125кб/с, а в настройках CAN задано 250кб/с, то после прошивки модуля любой программой, скорость его работы изменится на 250 кб/с.




Перед тем как создать функцию приема или передачи, необходимо добавить в программу новый идентификатор (ID), для этого перейдем в настройки ID

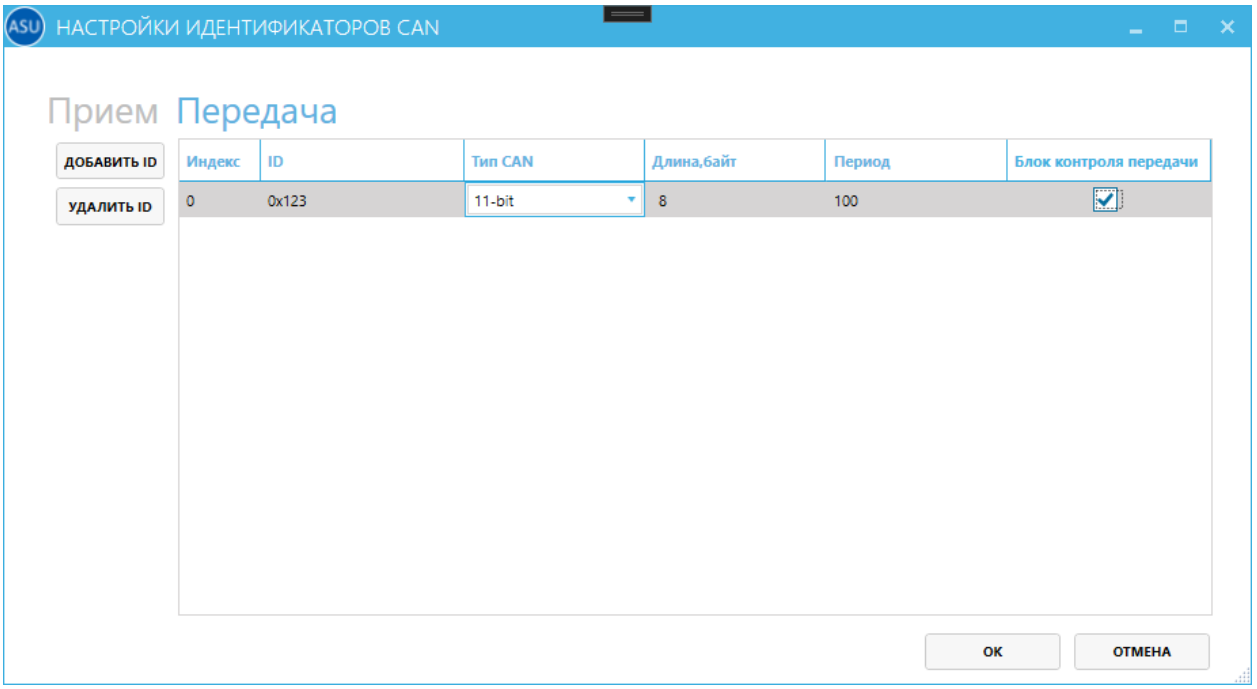



И создадим ID для функций приема, для этого зададим следующие параметры:



- Индекс – порядковый номер идентификатора в программе (может понадобиться при написании кода на языке Си)
- Идентификатор – ID функции
- Тип CAN – 11 или 29 бит
- Длина, байт – от 1 до 8
- Блок состояния – позволяет информировать проверить было ли принято какое-либо сообщение с указанным ID

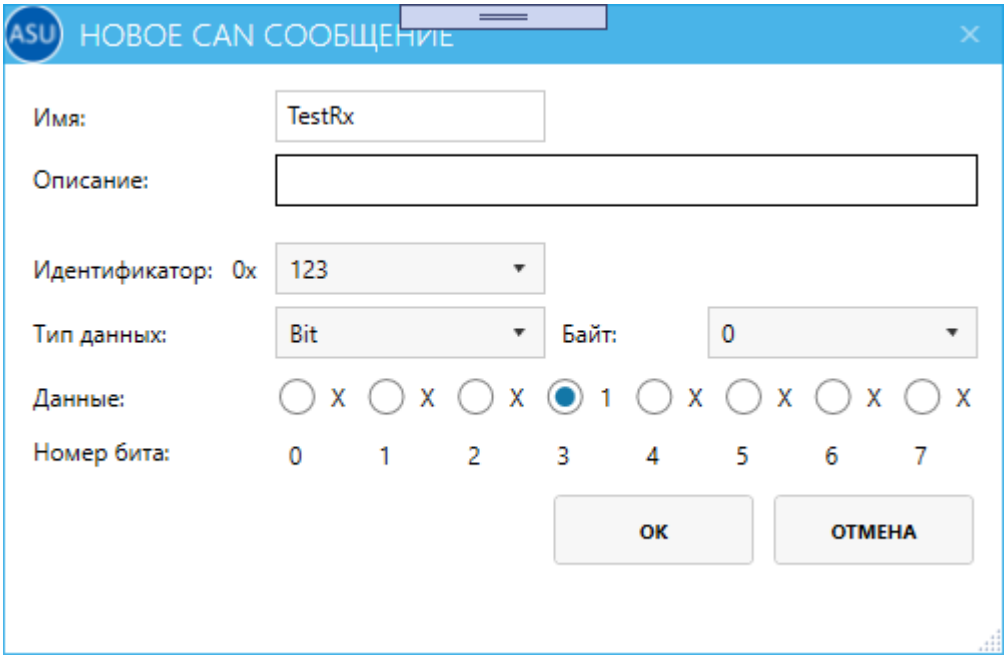
Параметры ID для функций передачи несколько отличаются:



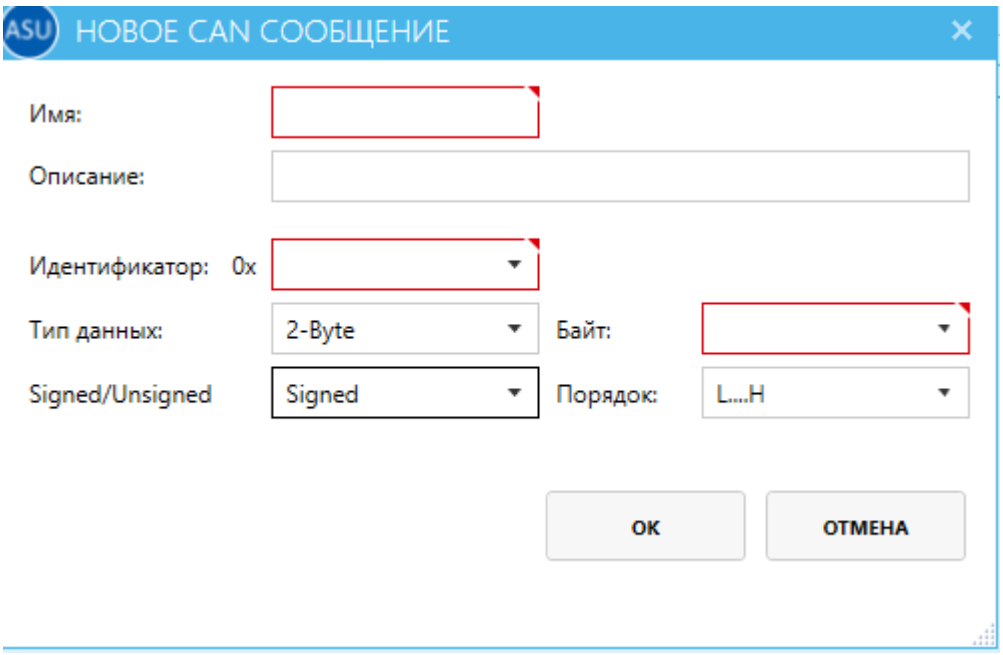

- Индекс – порядковый номер идентификатора в программе (может понадобиться при написании кода на языке Си)
- Идентификатор – ID функции
- Тип CAN – 11 или 29 бит
- Длина, байт – от 1 до 8
- Период – период отправки сообщения
- Блок контроля передачи – позволяет информировать принимающую сторону, что было отправлено сообщение с указанным ID.

После того, как мы создали идентификаторы для функций приема и передачи, мы можем приступить к созданию самих функций. Для этого нажмем кнопку «Создать» и заполним параметры в открывшемся окне:

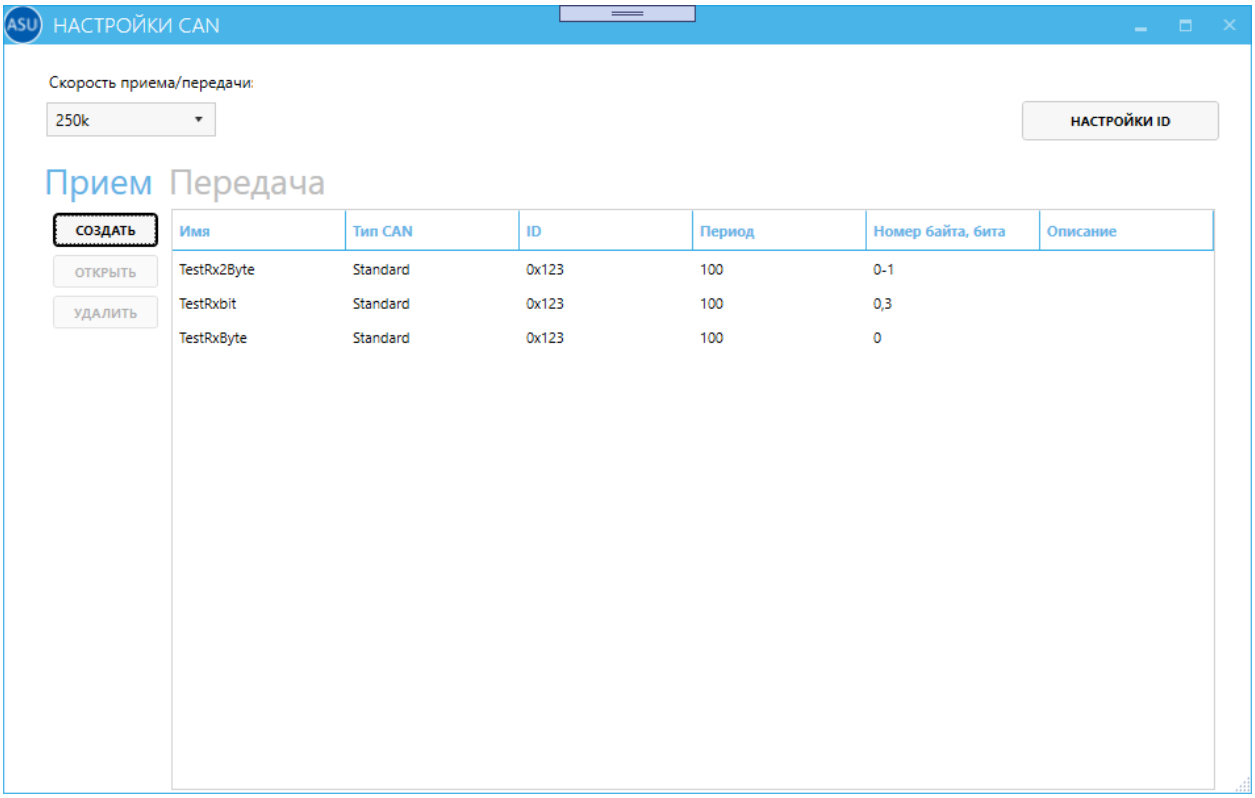
1. Имя функции (задается латиницей и не должно содержать специальных символов)
2. Описание функции
3. Идентификатор (выбирается из ранее созданных)
4. Тип передаваемых данных (Bit, Byte или 2-Byte).
5. Номер байта, в котором будут передаваться данные
6. Данные/номер бита (графа доступна только для типа данных Bit).

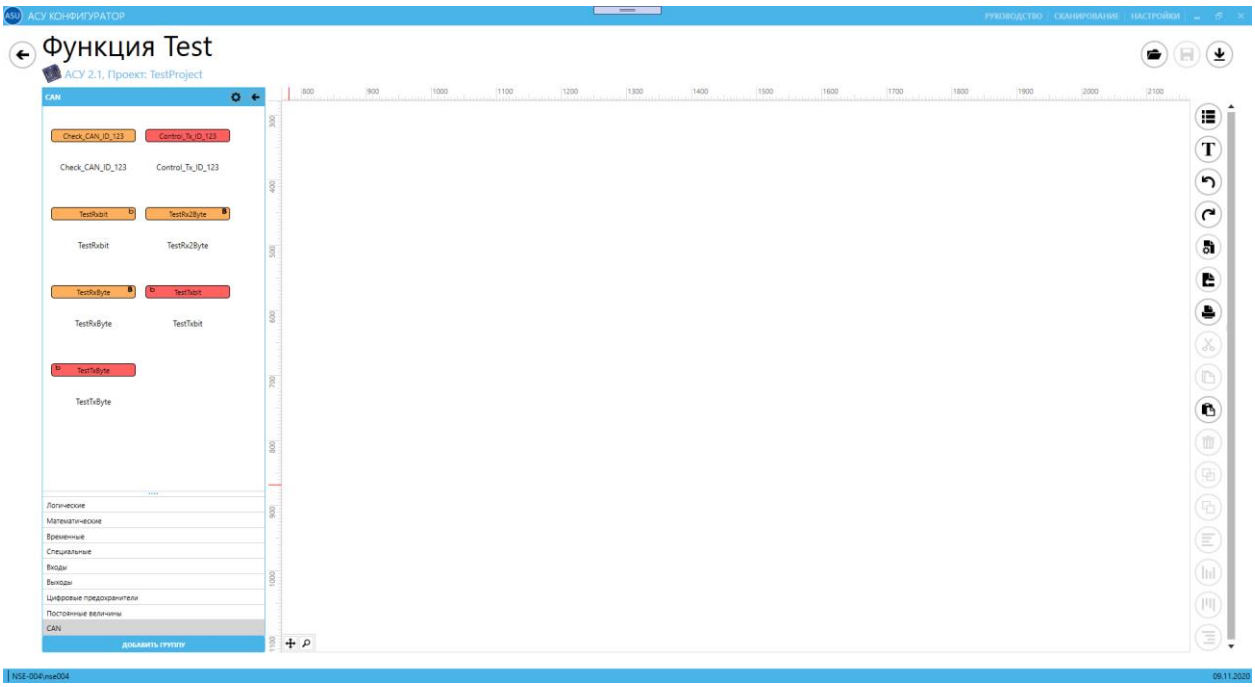



При выборе типа данных 2-Byte, появятся дополнительные графы с выбором порядка байт (от младшего к старшему или наоборот) и размерность данных (signed или unsigned).

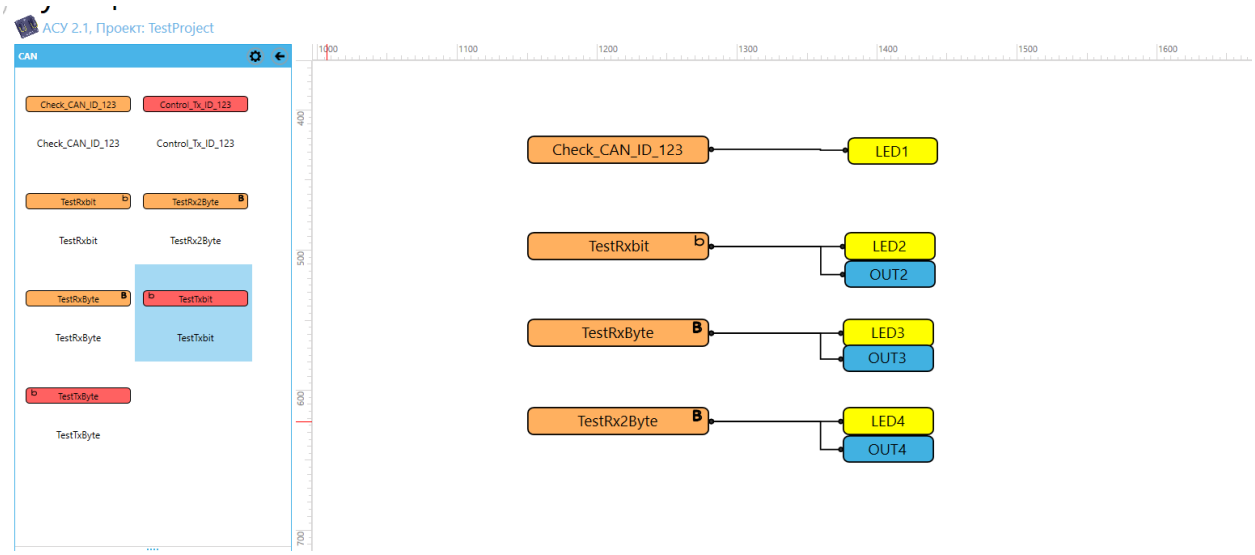



После создания функции появится в списке, а также в группе CAN

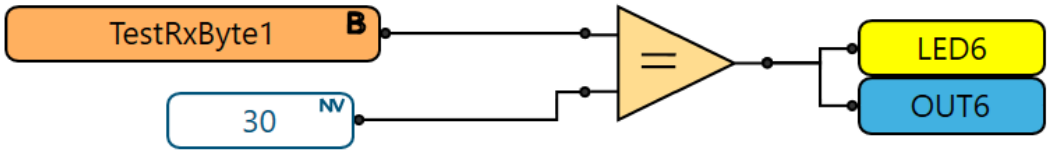




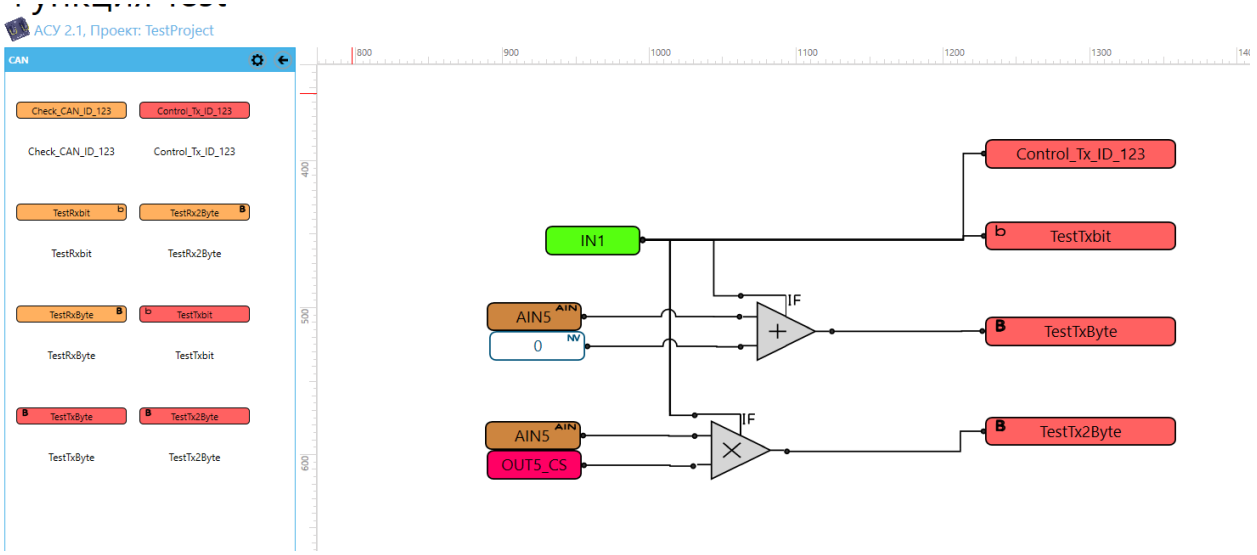

В приведенном ниже примере показана работа созданных ранее функций. Так, если по шине CAN получено сообщение, соответствующее третьему биту нулевого байта, загорится светодиод LED2 на модуле и включится выход OUT2, если получено сообщение, соответствующее полностью заполненному байту 0, загорится светодиод LED3 и включится выход OUT3, если получено сообщение, состоящее из 2х заполненных байт 0-1, то загорится светодиод № LED4 и включится выход OUT4. Проверка приема сообщений с ID 123 происходит путем вывода соответствующего сигнала на LED1.




Помимо проверки пришел ли заполненный полностью байт или конкретный бит в байте, мы можем проверять значение, передаваемое в байтовых сообщениях. Например, проверим, что значение из получаемого сообщения равно 30 (соответствует 1E в 16-ричном формате) при помощи блоков РАВНО и ЧИСЛО.



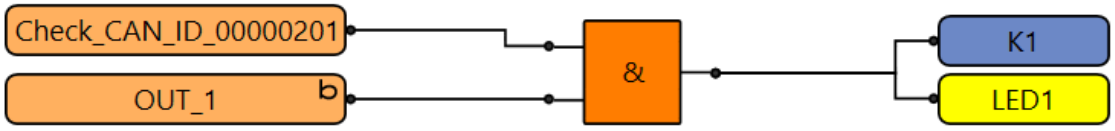

Пример использования функций отправки приведен ниже.



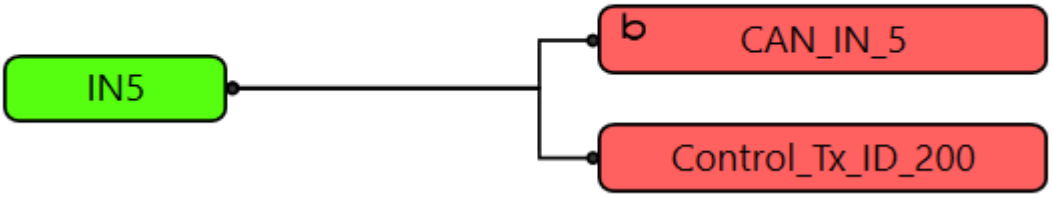

При данной конфигурации, после подачи сигнала на IN произойдет отправка бита #4 нулевого байта, сообщения контроля передачи, результата сложения аналогового сигнала на входе AIN5 и 0 (позволяет отправить значение AIN5 по условию наличия IN1 без изменений величины сигнала), а также результата перемножения величины напряжения на AIN5 и тока на OUT5.

Пример использования блоков состояния CAN и контроля передачи

Блок состояния CAN позволяет убедиться, что пришло хоть какое-то сообщение с заданным идентификатором, в том числе пустое. Благодаря использованию данного блока, мы, например, можем создать алгоритм, который бы гарантировал, что пришедшее сообщение не является сбоем из-за зависания CAN.




Аналогично с блоком передачи




В данном случае мы не только передаем само сообщение, но и информируем принимающую сторону, что была совершена новая отправка с указанным идентификатором.

Особенности использования блоков состояния CAN

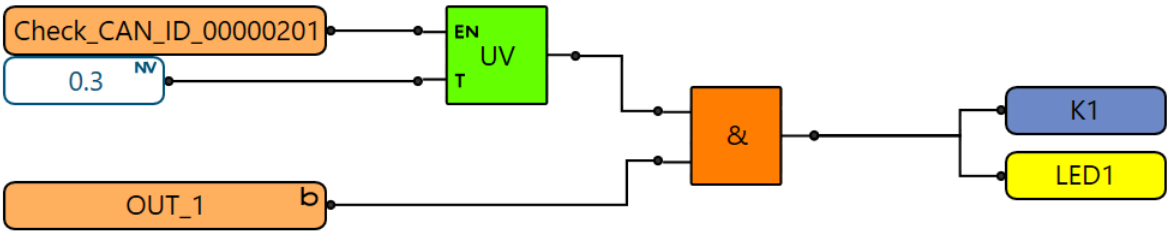
В случае если период отправки CAN сообщений превышает период выполнения цикла программы принимающего модуля (задается в окне функций), могут быть ложные срабатывания при использовании блоков состояния, связанные с необходимостью обнуления состояния сообщения в конце цикла.

Алгоритм работы CAN у модулей сейчас следующий: при приходе CAN-сообщения срабатывает прерывание, где данные из сообщения записываются в буфер и устанавливается флаг проверки блока состояния (Check ID). Данные по Chek ID и самому сообщению считываются в цикле программы. Для того, чтобы при следующем выполнении цикла не было ложных срабатываний нам нужно сбросить флаг Check ID. Единственным местом где это можно сделать является конец цикла программы. По умолчанию период выполнения функций 100мс. Таким образом Chek ID сбрасывается раз в 100мс, и в случае если сообщения приходят с периодом, превышающим 100мс, возможны ошибки в работе алгоритма и ложные срабатывания выходов, завязанных на этот блок.

Выходом из ситуации является:

1. Увеличение периода выполнения цикла программы до величины, превышающей период принимаемых сообщений.

2. Использование блоков задержки, например, одно вибратора с установленным значением, превышающим период принимаемых сообщений. Ниже показан пример, в котором сообщения блока состояния приходят с периодом 250мс, а программа выполняется с периодом 100мс. Таким образом для корректной работы нам нужно установить задержку >250мс, например, 300мс.

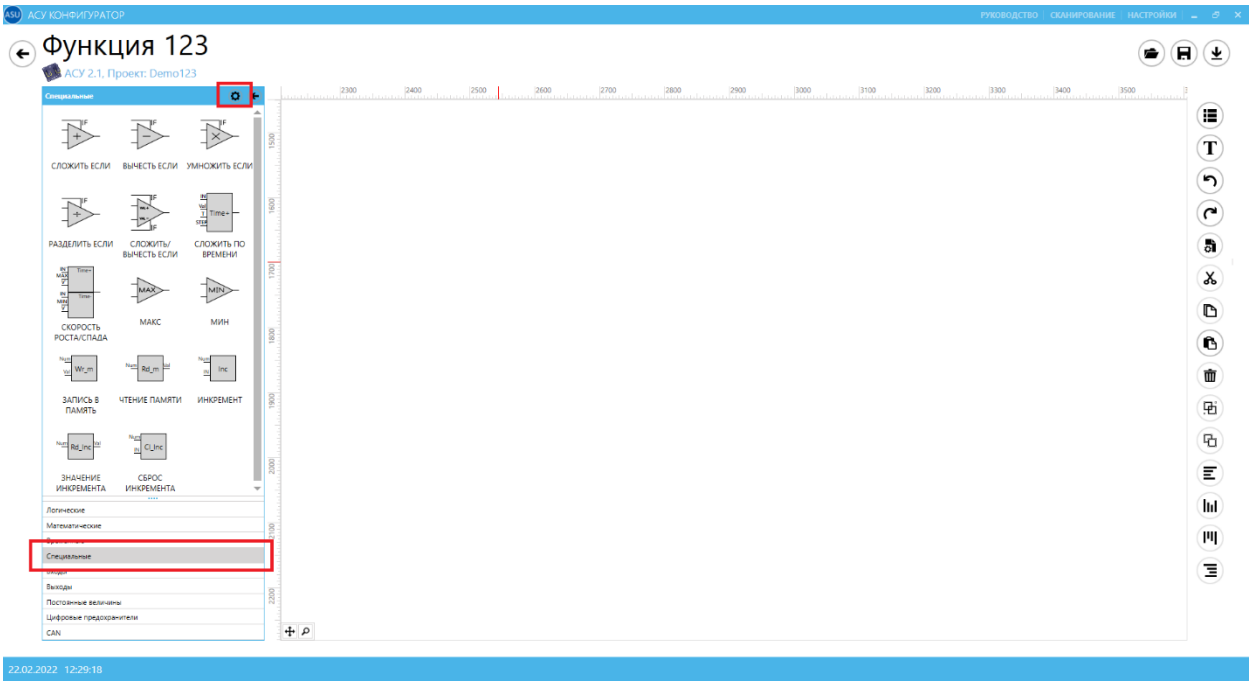



3. Уменьшить период принимаемых сообщений. В случае если за отправку сообщений отвечает другой блок АСУ, то достаточно зайти в настройки CAN ID и установить период передачи меньше, чем цикл работы программы на принимающем модуле.

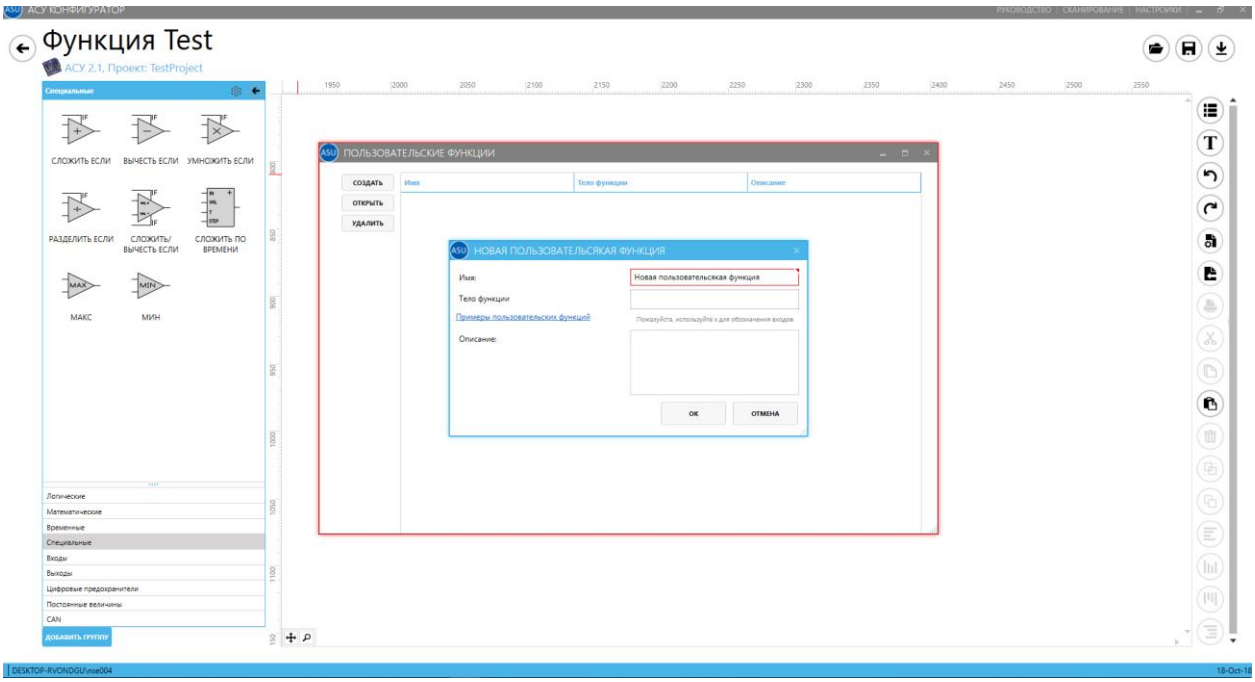
# Создание пользовательских функций

Пользовательские функции позволяют пользователю самому задавать алгоритм обработки сигнала, что бывает особенно полезно при корректировке аналоговых сигналов.

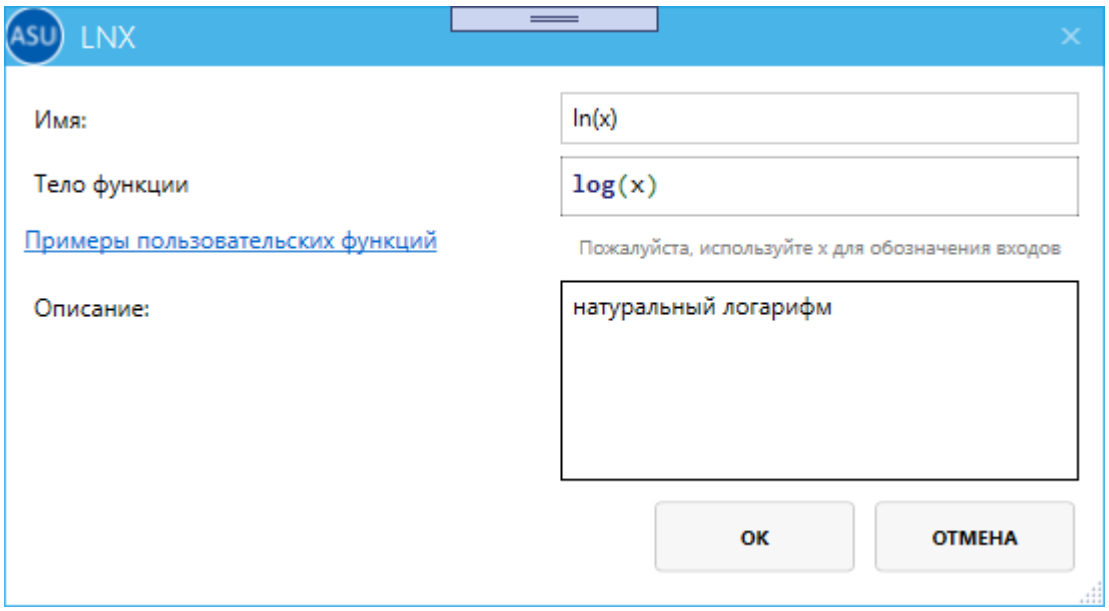
Для создания пользовательской функции необходимо перейти в группу «Специальные», открыть окно настроек.

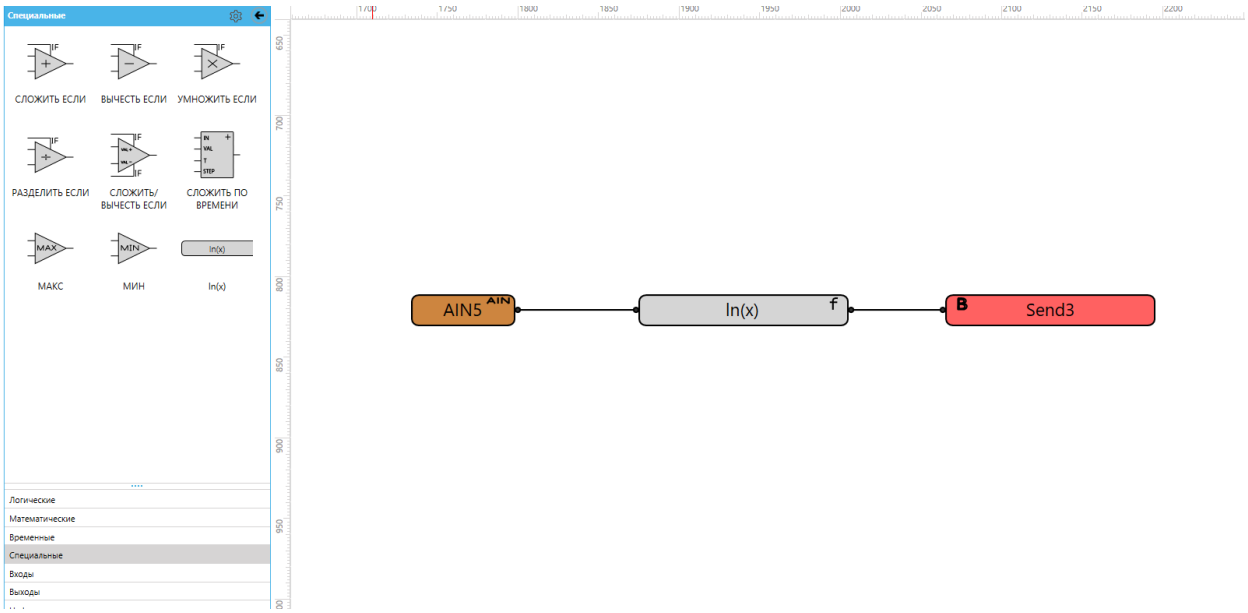



В открывшемся окне необходимо создать пользовательскую функцию, задав ее имя и тело функции, где в качестве обрабатываемого входного значения необходимо использовать «x».




Для примера создадим функцию обработки поступающего на аналоговый вход сигнала, взятие логарифма от значения и отправка результата в байте CAN-сообщения. Для этого создадим пользовательскую функцию ln(x), где x-входной параметр функции. Обратите внимание, что тело функции необходимо задавать в соответствии с параметрами библиотеки math языка СИ.






Некоторые математические функции на языке СИ:

- fabs(x) модуль числа x
- sqrt(x) квадратный корень из числа x
- sin(x) синус числа x (x в радианах)
- cos(x) косинус числа x (x в радианах)
- exp(x) вычисление ex
- log(x) натуральный логарифм числа x
- log10(x) десятичный логарифм числа x

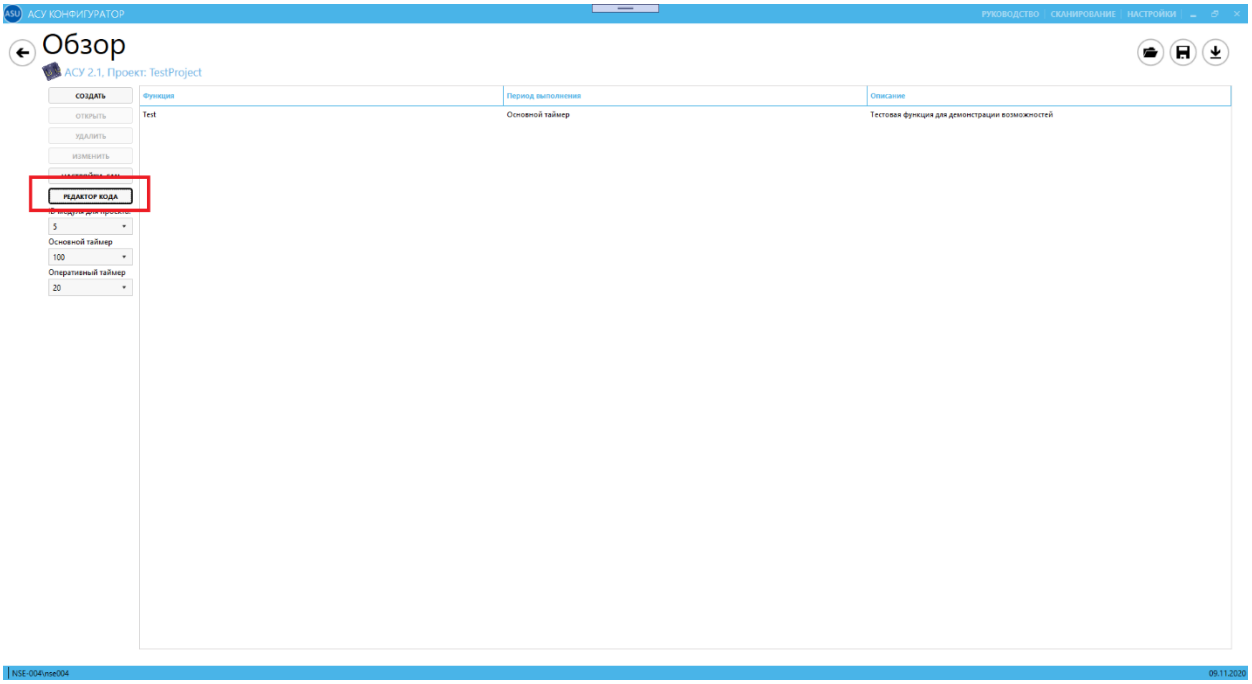
# Редактор кода

## Открытие редактора

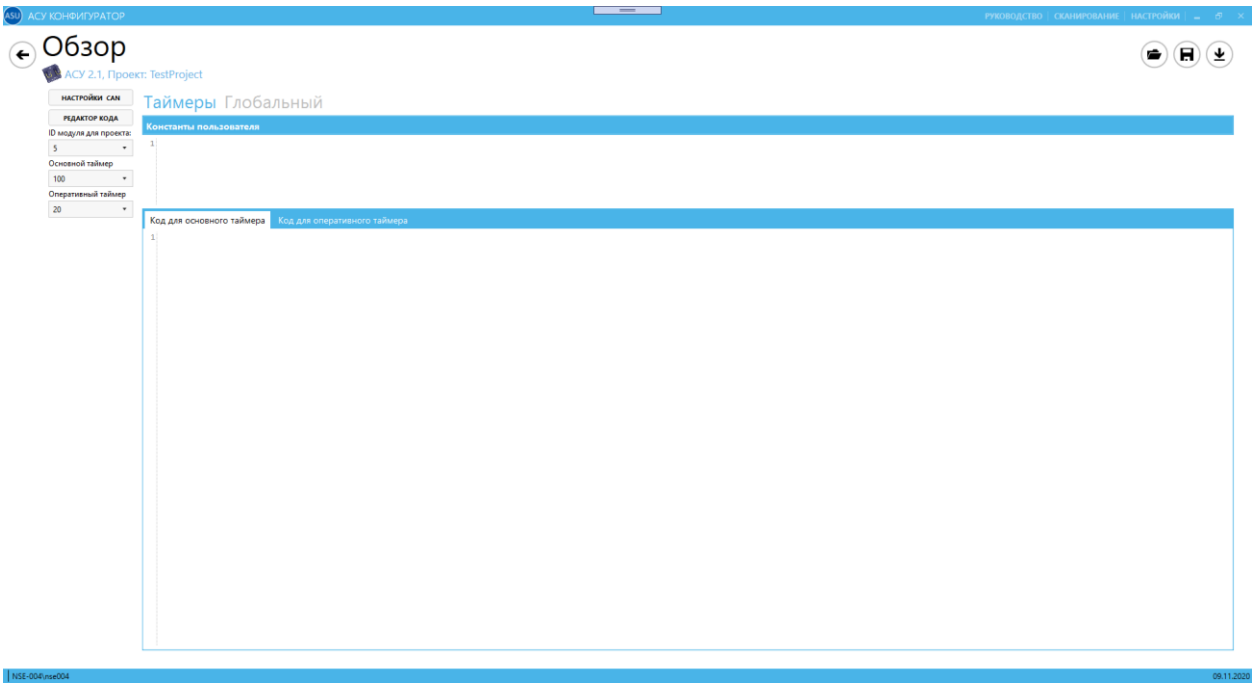
Иногда могут возникнуть ситуации, когда создания логических цепочек для работы модуля недостаточно или доступные компоненты не могут обеспечить требуемый функционал. В таком случае возможно применение редактора кода.

В данном редакторе пользователь может добавить в модуль отрезки собственного кода, написанного на языке СИ.

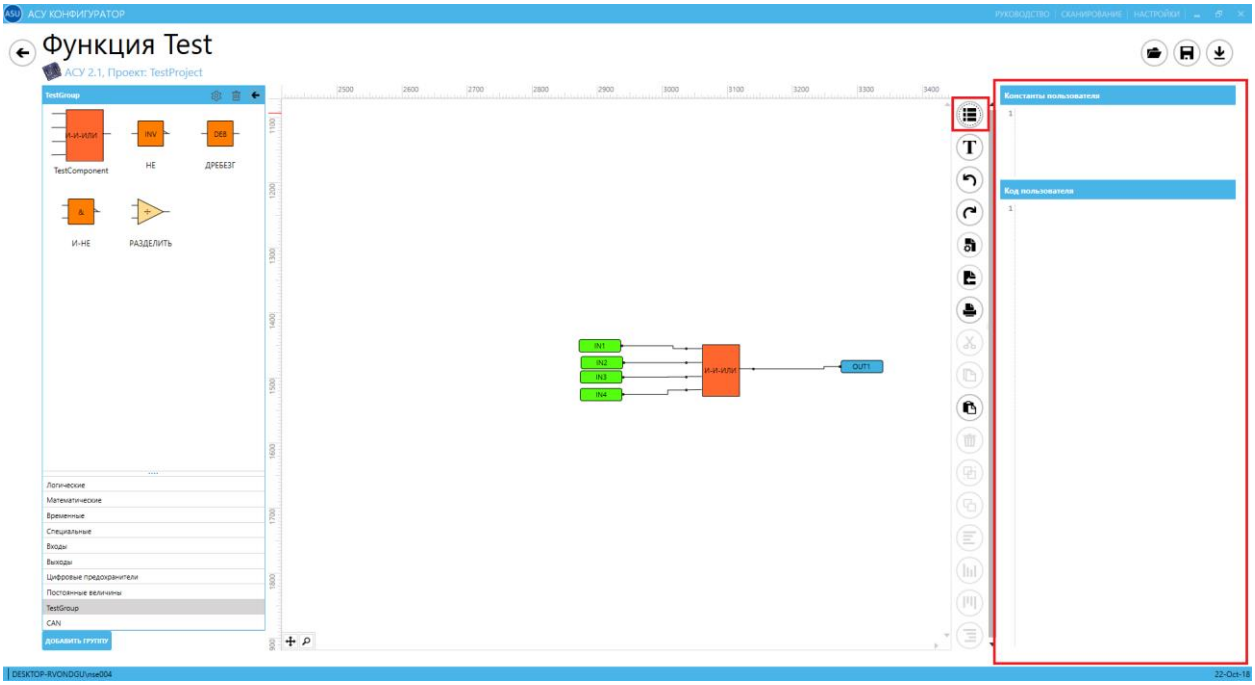
Редактор кода доступен в окне создания функций, а также в окне диаграммы.



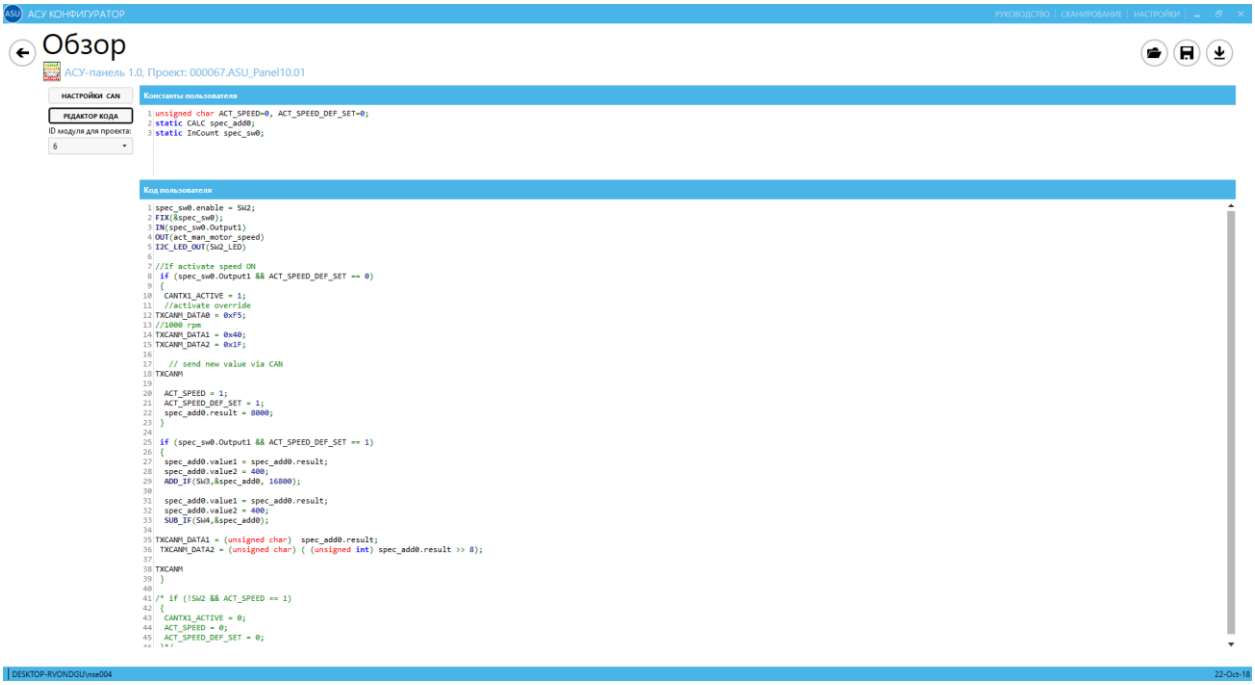

После открытия, пользователю доступны поля для работы с кодом для основного и оперативного таймера, а также глобального кода, который будет выполняться только 1 раз при запуске программы. Пользователь может изменять поля «Константы пользователя», куда записываются переменные, используемые в коде и «Код пользователя», куда записывается сам код.

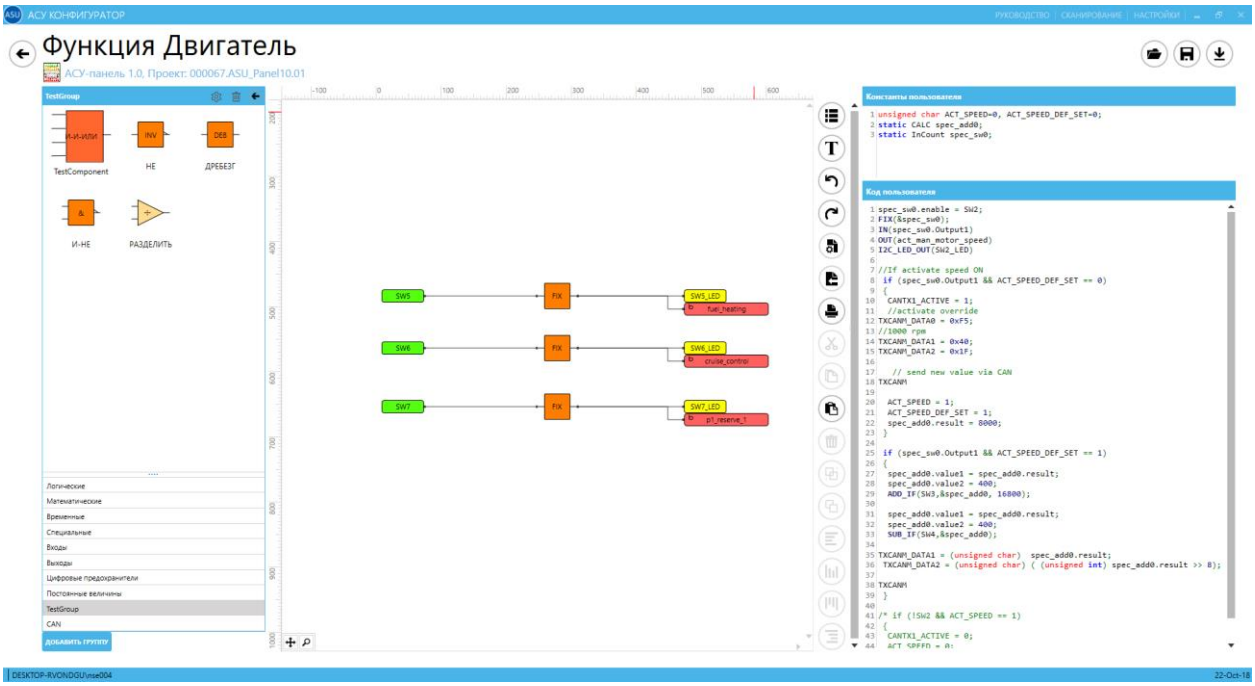



Для открытия в окне диаграммы, необходимо нажать соответствующую кнопку и справа появится поле для ввода и редактирования кода. Размер поля можно менять при помощи указателя мыши, предварительно наведя его на левую границу поля. Обратите внимание, что создаваемый код является общим для всех созданных функций.



Вид окна с примером кода представлен ниже.



## Создание кода

Переменные, задаваемые в первой графе, могут быть:

Числовые. Числовые переменные могут быть типа integer (int) или типа real (float). Integer (int) - это целые числа (например, 10 или -10). real (float) значения могут иметь десятичную точку. (Например, 1.23 или -20.123).

Символьные. Символьные переменные-это буквы алфавита, символы ASCII или цифры 0-9. Если вы объявляете символьную переменную, вы всегда должны помещать символ между одинарными кавычками (например, ’A’ ). Помните, что символ, не помещенный в кавычки не является таковым(например, 74 != ’74’, в данном случае 74 является числом, а ’74’-строкой).

Константы. Разница между переменными и константами заключается в том, что переменные могут изменять свое значение в любое время в зависимости от созданного алгоритма, константы же всегда сохраняют свое состояние. Константы могут быть очень полезны, например, Pi (число Пи) является хорошим примером использования константы.

Синтаксис работы с редактором кода полностью соответствует синтаксису языка СИ.

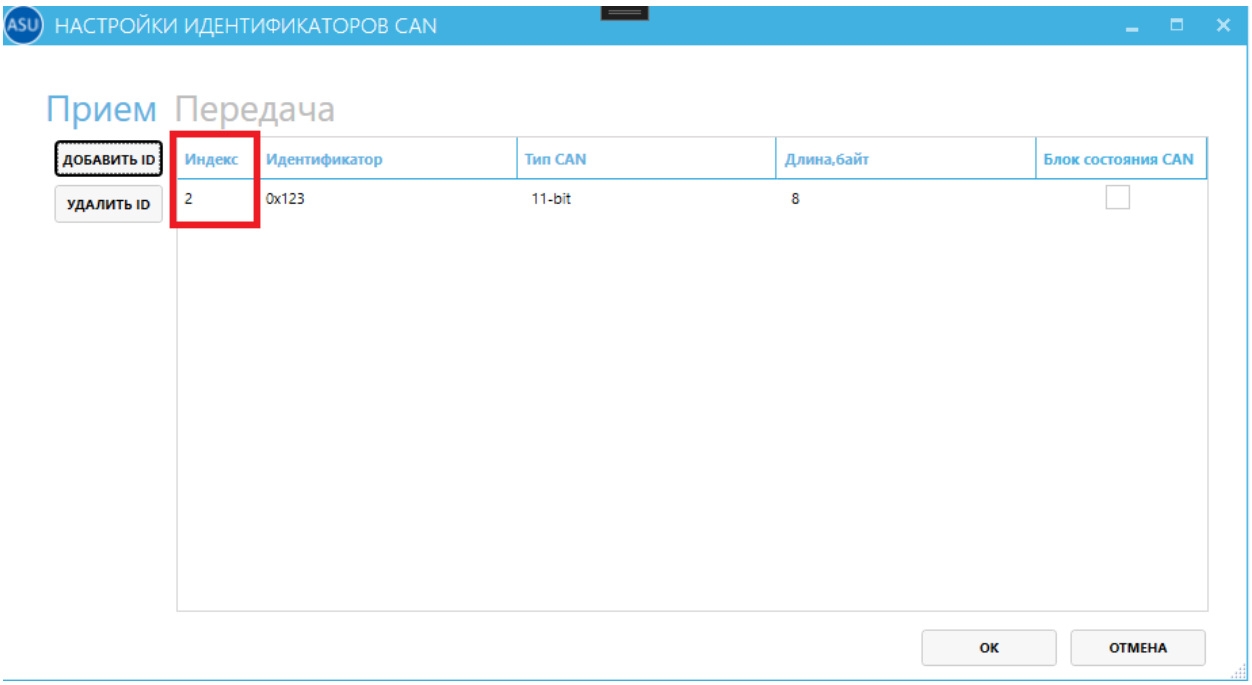
В таблице ниже приведены основные функции для работы с редактором кода

| Функция | Описание | Пример | Описание примера |
|---|---|---|---|
| IN() | Одна из основных функций, отвечает за проверку состояния входа, устанавливает значение глобальной переменной, которая будет записана в выход, аналогична условию «if» | IN(IN1)<br>OUT(OUT1) | При наличии высокого сигнала на IN1, на выход OUT1 будет подан высокий сигнал. Можно заменить выражением<br>if(IN1)<br>{<br>VS=1;<br>OUT(OUT1)<br>}<br>,где VS-глобальная переменная системы для установки состояния логических выходов. |
| OUT() | Используется для установки состояния на тот или иной выход | | |

| | | | |
|---|---|---|---|
| NOR() | Функция инверсии сигнала | IN(IN1)<br>NOR(IN1)<br>OUT(OUT1) | При наличии низкого сигнала на IN1, на выход OUT1 будет подан высокий сигнал и наоборот. Можно заменить выражением<br>if(IN1)<br>{<br>VS=0;<br>OUT(OUT1)<br>} |
| AND() | Функция «И» | IN(IN1)<br>AND(IN2)<br>OUT(OUT1) | При наличии высокого сигнала на IN1 и высокого сигнала на IN2 на выход OUT1 будет подан высокий сигнал. Можно заменить выражением<br>if(IN1&&IN2)<br>{<br>VS=1;<br>OUT(OUT1)<br>} |
| OR() | Функция «ИЛИ» | IN(IN1)<br>OR(IN2)<br>OUT(OUT1) | При наличии высокого сигнала на IN1 или высокого сигнала на IN2 на выход OUT1 будет подан высокий сигнал. Можно заменить выражением<br>if(IN1\|\|IN2)<br>{<br>VS=1;<br>OUT(OUT1)<br>} |
| PWM_OUTxx() | ШИМ выход, вместо «xx» необходимо указать номер выхода из доступных для ШИМ | PWM_OUT10(50) | Задает значение коэффициента заполнения для выхода номер 10 |
| CAN_initRxObj(x1,x2,x3,x4); | Инициализация функции приема сообщения для указанного ID, где x1- индекс ID в списке, x2-разрядность идентификатора (0-11bit, 1-29bit), x3-идентификатор в формате 0x000, x4- всегда задается как 0, величина на данный момент не используется. | CAN_initRxObj(8, 1, 0x18FE1383, 0);<br><br>CAN_initRxObj(4, 0, 0x027, 0) | В первом варианте показана инициализация для сообщений с ID 0x18FE1383, который соответствует индексу 8 и имеет разрядность 29bit.<br><br>Во втором варианте показана инициализация для сообщений с ID 0x027, который соответствует индексу 4 и имеет разрядность 11bit. |
| CAN_initTxObj(x1,x2,x3,x4, x5, x6); | Инициализация функции отправки сообщения для указанного ID, где x1- индекс ID в списке, x2-разрядность идентификатора (0-11bit, 1-29bit), x3-идентификатор в формате 0x000, x4- всегда | CAN_initTxObj(5, 1, 0x18FE4385, 1, 8, 100);<br><br>CAN_initTxObj(0, 0, 0x025, 1, 8, 100); | В первом варианте показана инициализация для сообщений с ID 0x18FE4385, который соответствует индексу 5 и имеет разрядность 29bit, период отправки 100 мс<br><br>Во втором варианте показана инициализация для сообщений с ID 0x027, который соответствует |

| | | | |
|---|---|---|---|
| | задается как 1, величина на данный момент не используется, x5- количество байт данных в посылке CAN, от 1 до 8, x6-период отправки сообщения. | | индексу 4 и имеет разрядность 11bit, период отправки 100 мс |
| CAN_RXNb(x1, x2,x3,x4) | Прием бита сообщения, где x1-индекс ID в списке, x2-номер байта, x3-номер бита, x4-длина сообщения (для битовых сообщений она равна 1) | if(CAN_RXNb(2,0,3,1))<br>{<br>IN(1)<br>OUT(OUT1)<br>} | Если пришло сообщение с ID, соответствующему индексу 2 в списке сообщений (смотри в настройках ID раздела CAN) и в этом сообщении в байте номер 0, бит номер 3 задан 1, то на выходе OUT1 будет высокий сигнал |
| CAN_RX1B(x1, x2) | Прием байта сообщения, где x1-индекс ID в списке, x2-номер байта | if(CAN_RX1B(2,0))<br>{<br>IN(1)<br>OUT(OUT1)<br>} | Если пришло сообщение с ID, соответствующему индексу 2 в списке сообщений и в этом сообщении заполнен байт номер 0, то на выходе OUT1 будет высокий сигнал |
| CAN_TXNb(x1, x2,x3,x4,x5) | Передача бита сообщения, где x1- индекс ID в списке, x2-номер байта, x3-номер бита, x4-длина сообщения (для битовых сообщений она равна 1), x5 всегда задается как 1, величина на данный момент не используется | if(IN1)<br>{<br>CAN_TXNb(0,1,4,1,1)<br>} | При наличии высокого сигнала на IN в шину CAN будет отправлено сообщение с ID, соответствующему индексу 2 в списке сообщений, байт №1, бит №4 |
| CAN_TX1B(x1, x2,x3) | Передача байта сообщения, где x1- индекс ID в списке, x2-номер байта, x3-значение, записываемое в байт (в десятичной или шестнадцатеричной системе), для указания значения в шестнадцатеричной системе необходимо добавить перед ним 0x без кавычек. | if(IN1)<br>{<br>CAN_TX1B(0,0,AIN5)<br>}<br><br>CAN_TX1B(0,3,0xFF) | При наличии высокого сигнала на IN в шину CAN будет отправлено сообщение с ID, соответствующему индексу 0 в списке сообщений, байт №0 со значением напряжения на входе AIN5.<br><br>Второй пример показывает отправку фикции с записанным в байт №0 шестнадцатеричным значением. |

Обратите внимание, что при работе с CAN-сообщениями в качестве принимаемого и передаваемого параметра используется не ID как таковой, а его индекс в списке созданных.




Также стоит отметить, что функции инициализации CAN_initRxObj(x1, x2, x3, x4) и CAN_initTxObj(x1, x2, x3, x4, x5, x6) необходимо использовать только в том случае, если с указанным идентификатором не было создано ни одного сообщения средствами Конфигуратора. Лучше всего эти функции располагать в глобальном таймере инициализации.

Таймеры Глобальный

Глобальные константы

```
1
```

Код для таймера инициализации

```
1 CAN_initTxObj(0, 1, 0x10011000, 1, 8, 100); //инициализация сообщений на отправку
2
```

Пример работы с выходами модуля

Установка светодиода LED1 в постоянное состояние «Включен»

Код для основного таймера | Код д

```
1 VS=1;
2 OUT(LED1);
3
4
```

Установка состояния светодиода в зависимости от входа IN1

Код для основного таймера Код

```
if(IN1)
{
VS=1;
OUT(LED1);
}

if(!IN1)
{
VS=0;
OUT(LED1);
}
```

Установку состояния также можно проводить без использования системной переменной VS, используя вместо нее функцию IN(), например

Код для основного таймера К

```
if(IN1)
{
IN(1)
OUT(LED1);
}

if(!IN1)
{
IN(0)
OUT(LED1);
}
```

Пример работы с CAN-сообщениями

Редактор кода позволяет работать как напрямую с CAN-функциями, например, так и переопределять их имя для более простого использования в коде.

*Пример использования CAN-функции в коде*

мс

Код для основного таймера | Код для оперативного таймера

```
1 if(CAN_RXNb(2,0,3,1))
2 {
3 IN(1)
4 OUT(OUT1)
5 }
6
```

*Пример использования CAN-функции с заранее заданным именем*

Таймеры Глобальный

**Константы пользователя**

```
1 #define RxTest CAN_RXNb(2,0,3,1)
```

Код для основного таймера | Код для оперативного таймера

```
1 if(RxTest)
2 {
3 IN(1)
4 OUT(OUT1)
5 }
6
```

Пример проверки значения CAN-функции

Таймеры Глобальный

**Константы пользователя**

```
1 #define RxByteTest CAN_RX1B(2,0)
```

Код для основного таймера | Код для оперативного таймера

```
1 if(RxByteTest==30)
2 {
3 IN(1)
4 OUT(OUT1)
5 }
6
```

Описание: Если в значение, пришедшее в 0 байте сообщения с ID, соответствующем индексу 2 = 30, загорится первый светодиод.

Пример объявления и использования функции для сравнения двух величин

Таймеры Глобальный

Константы пользователя

```
1 int less_then_result;
```

Код для основного таймера | Код для оперативного тайме

```
1  void LessThanFunc(float val1, float val2)
2  {
3      if (val1 < val2) less_then_result = 1;
4      else  less_then_result = 0;
5  }
6
7  LessThanFunc(AIN5, AIN6);
8
9  if(less_then_result==1)
10 {
11 IN(1)
12 OUT(OUT1)
   }
```

*Описание:* на 1-5 строчках кода создадим функцию для сравнения двух величин типа float, результат будем выводить в константу less_than_result.

На строке 7 вызовем функцию и проверим значения аналоговых входов AIN5 и AIN6.

На строке 9 считаем значение less_than_result и если оно равно 1, то есть значение на AIN5 < AIN6, установим на OUT1 высокий сигнал.

Пример использования цикла

Таймеры Глобальный

Константы пользователя

```
#define TxTest  CAN_TXNb(0,1,4,1,1)
int less_than_result;
```

Код для основного таймера | Код для оперативного т

```
void LessThanFunc(float val1, float val2)
{
    if (val1 < val2) less_than_result = 1;
    else  less_than_result = 0;
}

while(less_than_result==0)
{
LessThanFunc(AIN5, AIN6);
TxTest;
}
```

В данном примере мы переопределили функцию отправки CAN-сообщения как TxTest и используем цикл while для проверки значения переменной less_than_result. Внутри цикла мы вызываем функцию сравнения значений аналоговых входов и отправляем сообщение CAN. Т.е. до тех пор пока AIN5 > AIN6 в шину CAN будет слаться сообщение с ID, соответствующему индексу 0 в программе, байт №1, бит №4.

При использовании редактора кода стоит быть внимательным и не использовать имена CAN-сообщений для переопределения, если такое имя уже используется в одном из сообщений, созданных средствами АСУ Конфигуратора.

Также следует избегать ситуаций, когда мы считываем значение аналогового входа и в то же время используем этот вход как цифровой.

При использовании ШИМ-выходов, не забудьте задать частоту их работы через настройки конфигуратора.

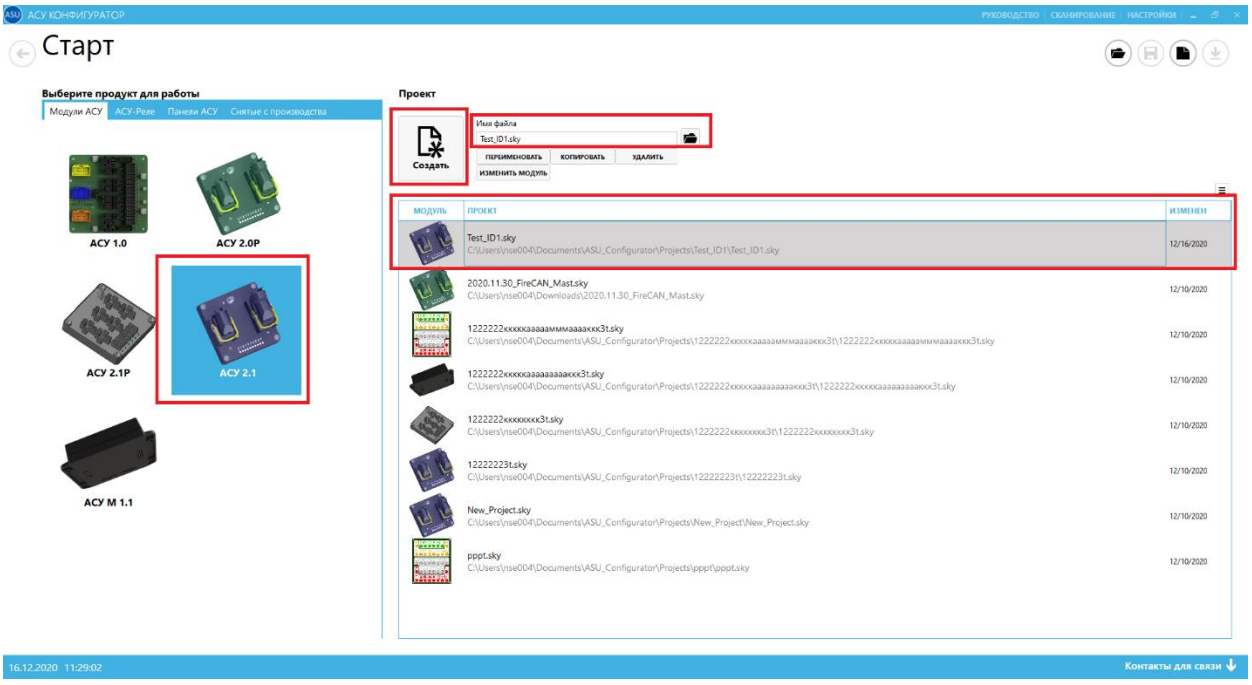
Обращаем также ваше внимание, что присутствует риск совпадения имен переменных или функций с теми, которые использует компания НСЭ. В будущем мы добавим проверку на совпадение имен, однако до тех пор старайтесь называть переменные и функции не совсем стандартными названиями, например, добавляя цифрой идентификатор в конец имени.

# Пример совместной работы нескольких модулей в шине CAN

В тех случаях, когда требуется организовать совместную работу нескольких модулей в рамках одной системы, наиболее оптимальным вариантом будет создание для каждого модуля отдельного проекта.

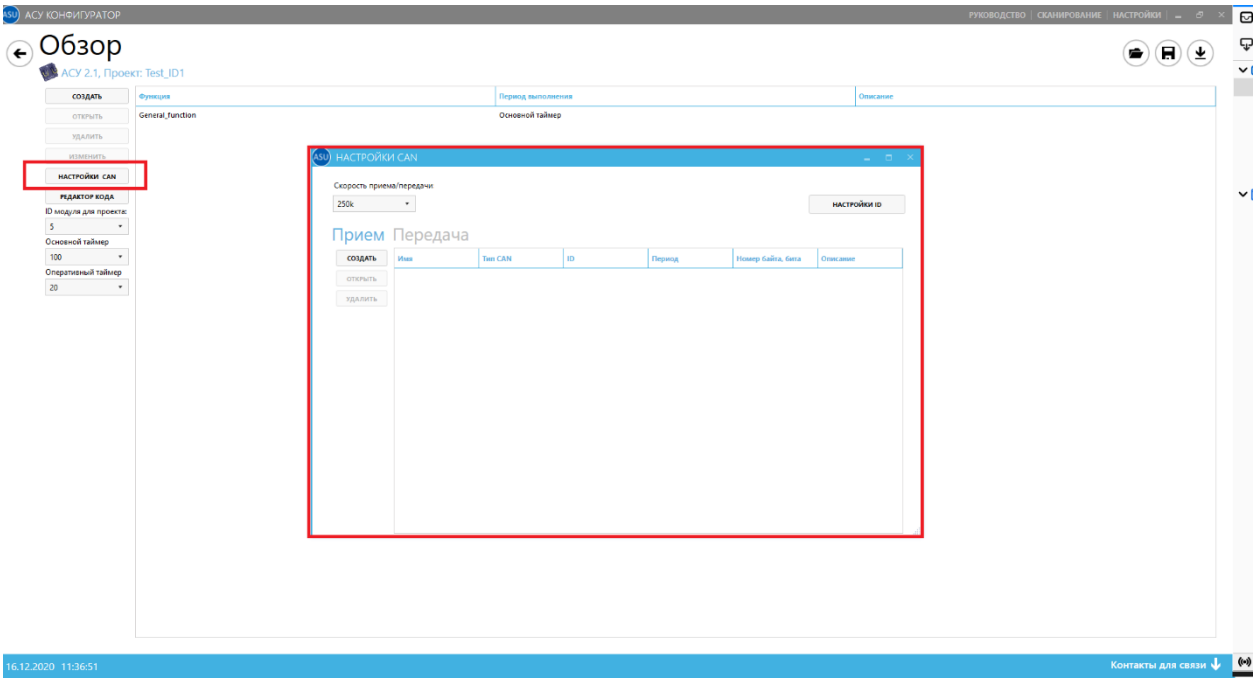
Рассмотрим пример совместной работы трех модулей АСУ 2.1. Первый модуль будет управлять включением выходов на двух других модулях и получать сведения о их работе, в том числе сигнал о наличии неисправности (обрыв или короткое замыкание). Второй и третий модуль получают сигнал управления по шине CAN от первого, активируют выходы и отправляют сообщение о их работе обратно на управляющий модуль.

Для начала создадим проект основного модуля: выберем модуль АСУ 2.1, зададим имя проекта как Test_M1 и нажмем кнопку «Создать».




Далее в проекте создадим функцию, в которой в последствии будем задавать алгоритм работы модуля.

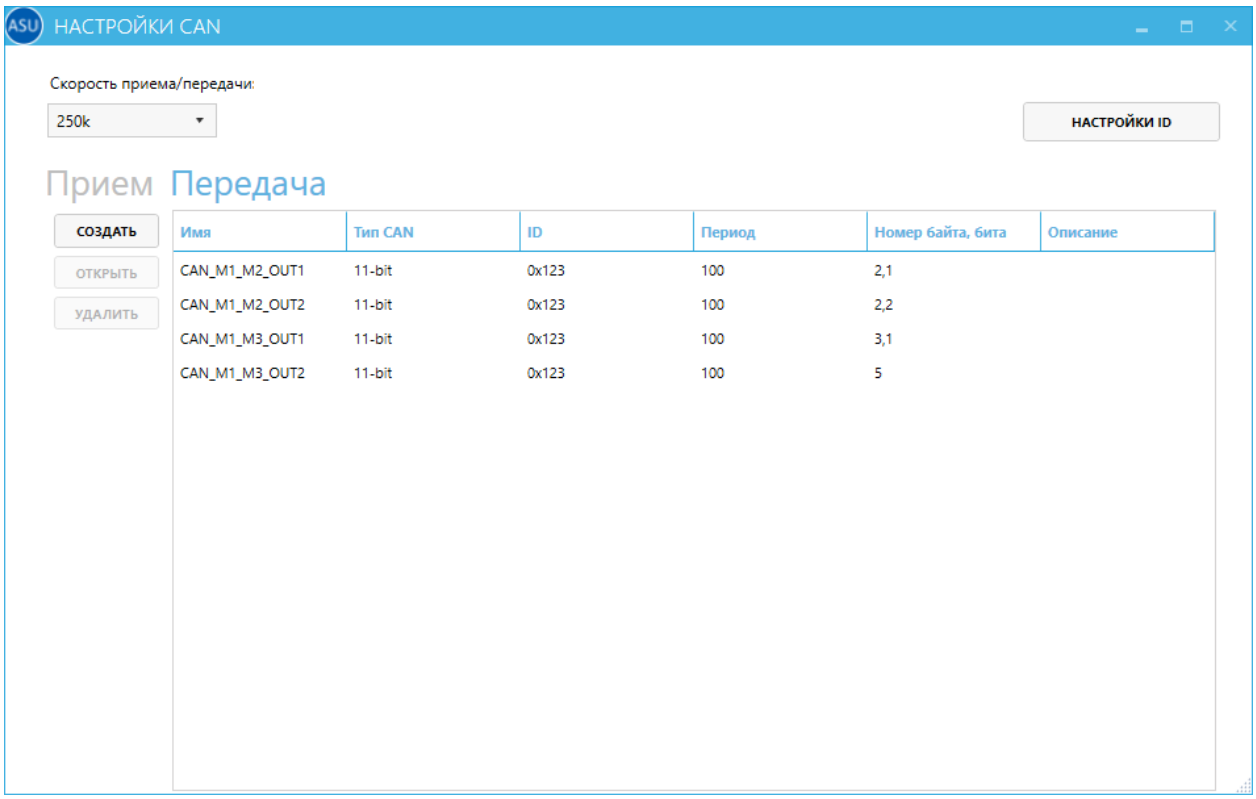

И сразу же перейдем в настройки CAN для создания сообщений приема и передачи.

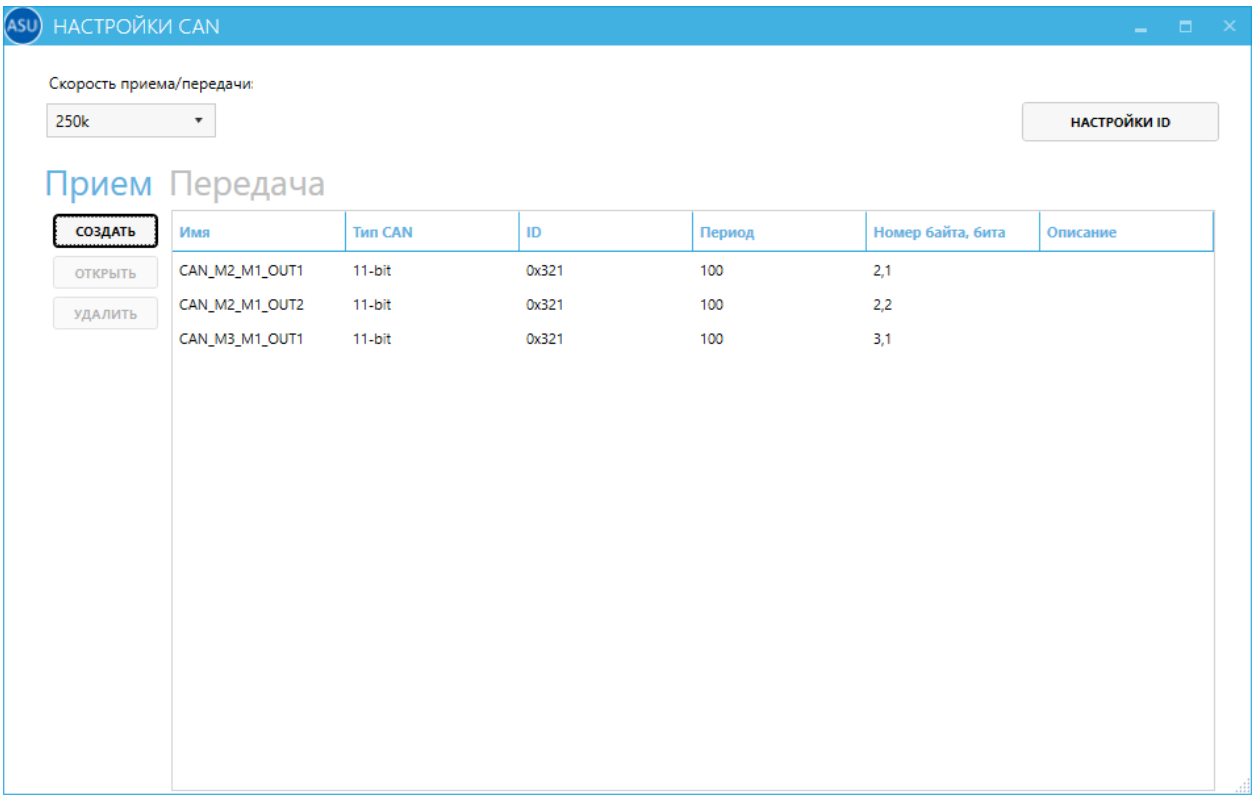



Предположим, что нам необходимо будет активировать 2 выхода на каждом из подключенных к сети модулей (всего 4 выхода) и получить обратный сигнал от 3 из них. Для этого нам потребуется создать 4 сообщения на передачу и 3 на прием.

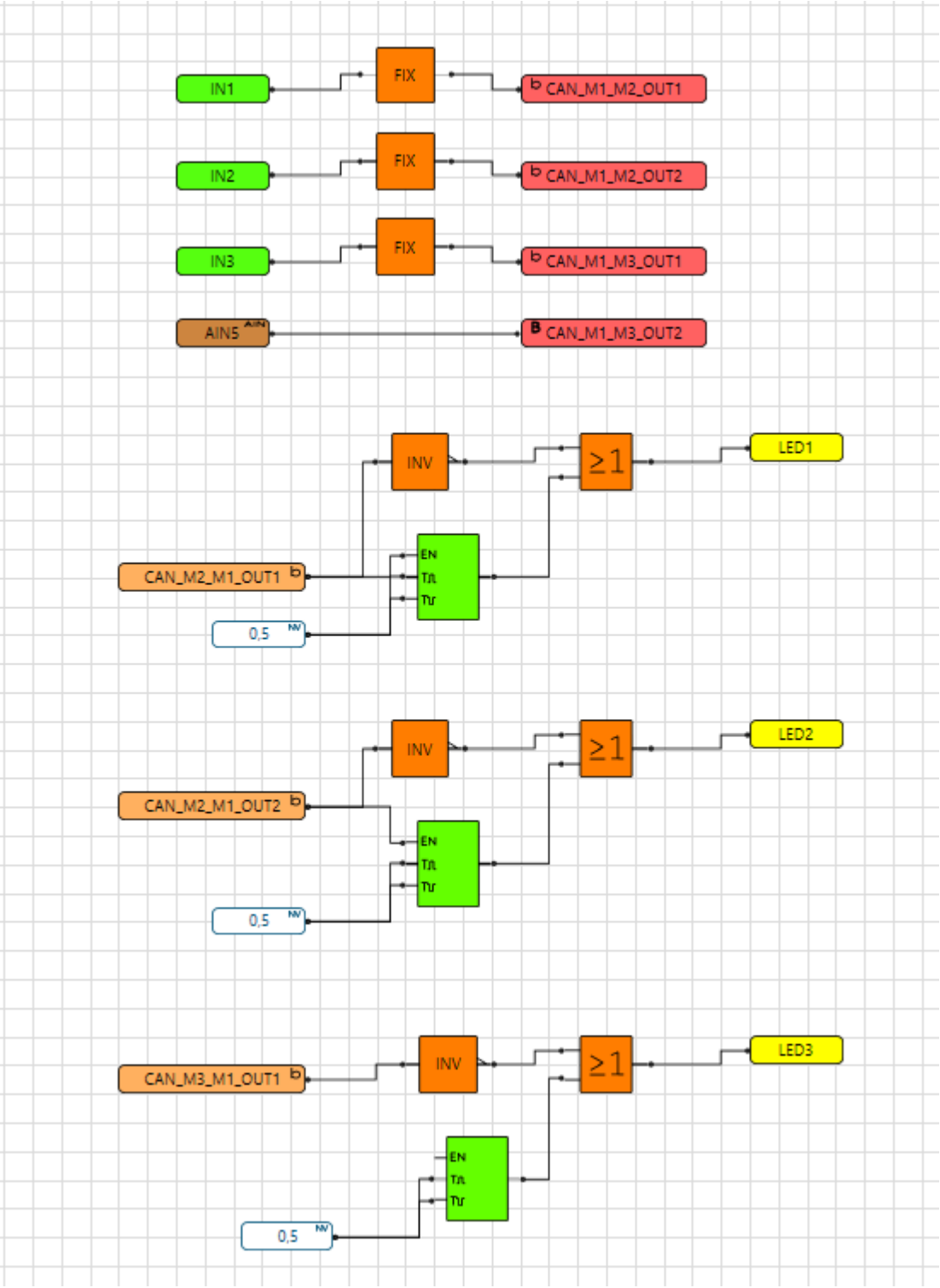
Создадим сообщения в соответствии с инструкцией, описанной в предыдущих разделах: создадим ID для приема и передачи и сообщения, соответствующие описанному ранее алгоритму (4 на передачу и 3 на прием). Сообщения всегда можно отредактировать или создать новые, поэтому не старайтесь выполнить настройку всех их сразу: когда алгоритм содержит большое количество операций, это может сложной задачей, поэтому лучше все делать по шагам с предварительной проверкой работоспособности.

При создании сообщений для упрощения идентификации были использованы имена CAN_M*_M**, где M*-передающий модуль, а M**-принимающий. Все сообщения имеют битовый формат за исключением CAN_M1_M3_OUT2, которое будет использоваться для передачи значения напряжения на одном из входов модуля.






После создания сообщений перейдем к построению схемы работы первого модуля.

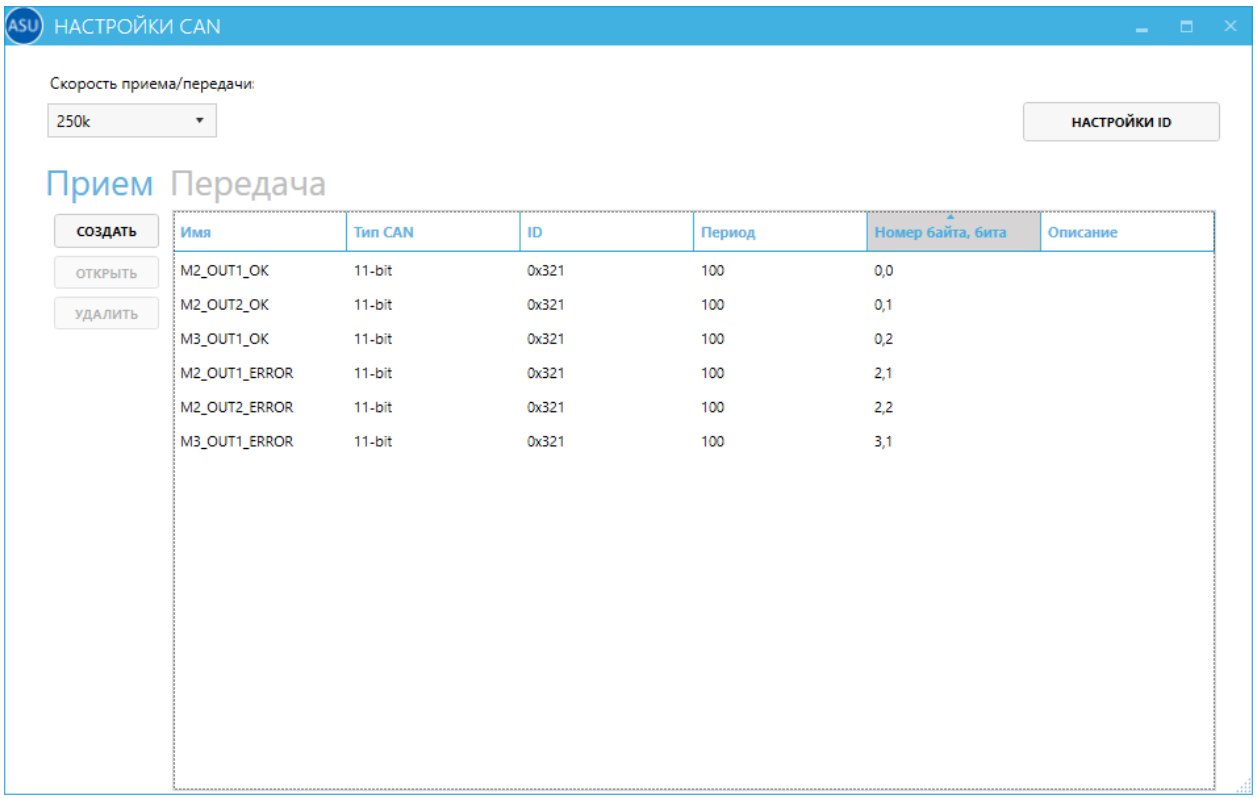



В соответствии с данным алгоритмом, при подаче сигнала на входы IN1-IN3 будет происходить отправка соответствующего CAN-сообщения на модули М2, М3 для активации их выходов. При повторной подаче сигнала на IN1-IN3

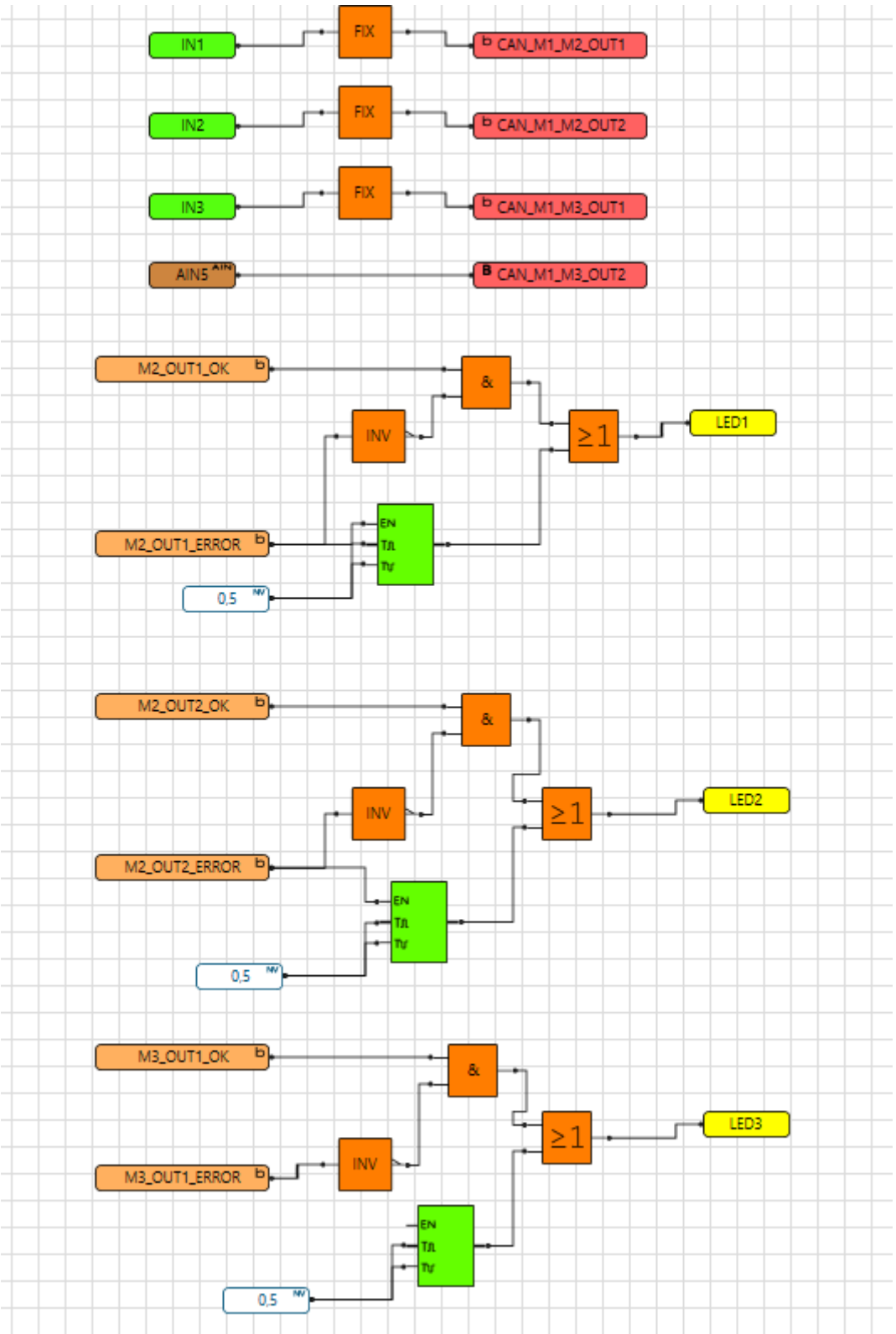
отправка сообщений будет остановлена (за это отвечает блок FIX-фиксация). На модуль М3 в режиме реального времени также будет передаваться значение напряжение с аналогового входа AIN5.

Диагностика работы выходов модулей М2 и М3 будет осуществляться за счет приема сообщений CAN_M2_M1_OUT1, CAN_M2_M1_OUT2 и CAN_M3_M1_OUT1. Данные сообщения свидетельствуют о наличии ошибки на выходе (КЗ или обрыв цепи). За индикацию отвечают светодиоды LED1-LED3 на борту модуля. По умолчанию светодиоды находятся в активном состоянии, но как только поступит сообщение об ошибке, они начнут мигать с интервалом 1 секунда (0.5 секунды включен, 0.5 секунды выключен). За режим работы диодов отвечают блоки ИЛИ и НЕ, а также блок ИМПУЛЬС. Данный алгоритм имеет один недостаток – в случае полного выхода модуля М2 или М3 из строя или обрыва сети CAN, модуль М1 будет по-прежнему указывать, что с выходами модулей все в порядке. Данную проблему можно решить добавлением еще трех сообщений на примем, как показано ниже.

Переименуем существующие сообщения на прием в M*_OUT*_ERROR и добавим 3 новых сообщения M*_OUT*_OK



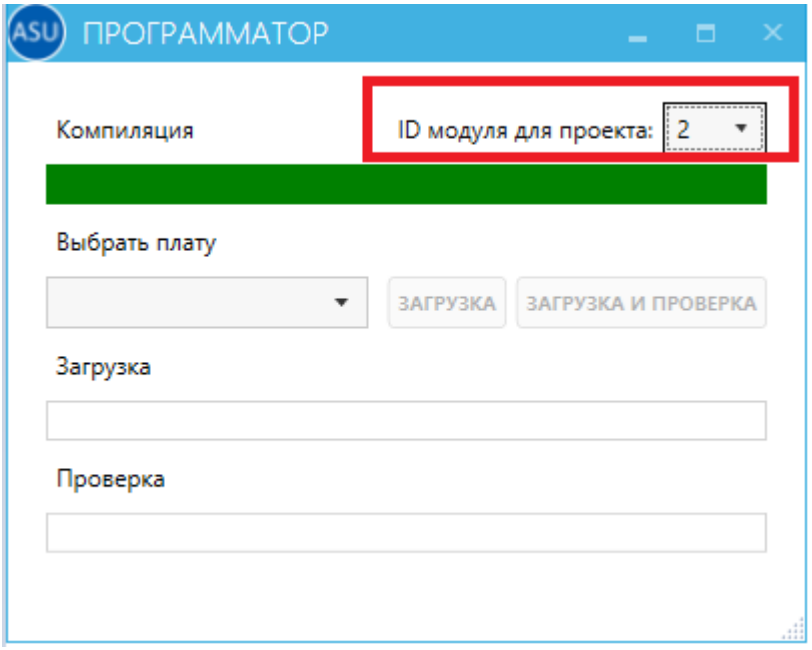

И отредактируем нашу схему.



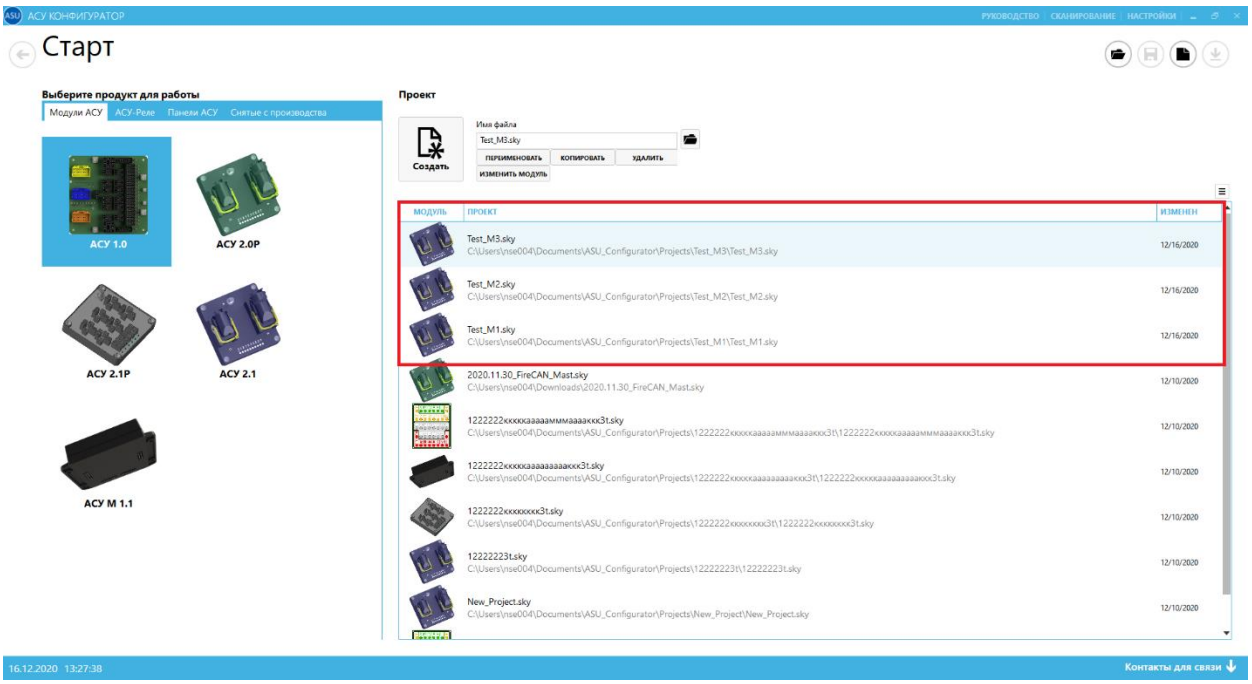

В соответствии с изменённым алгоритмом LED1-LED3 будут гореть только при наличии сообщений M*_OUT*_OK и мигать при приходе M*_OUT*_ERROR.

Такой подход повышает надежность системы, однако излишне нагружает шину CAN, что может сказаться при включении в нее большого количества устройств (сообщения могут доставляться с задержкой или вообще не приходить на модуль). При большой загруженности шины следует помнить о возможности изменения ID сообщений (чем меньше ID, тем выше приоритет) и периода отправки. В случае конфликта одним из простейших выходов является установка нестандартного периода передачи, например, 17мс.

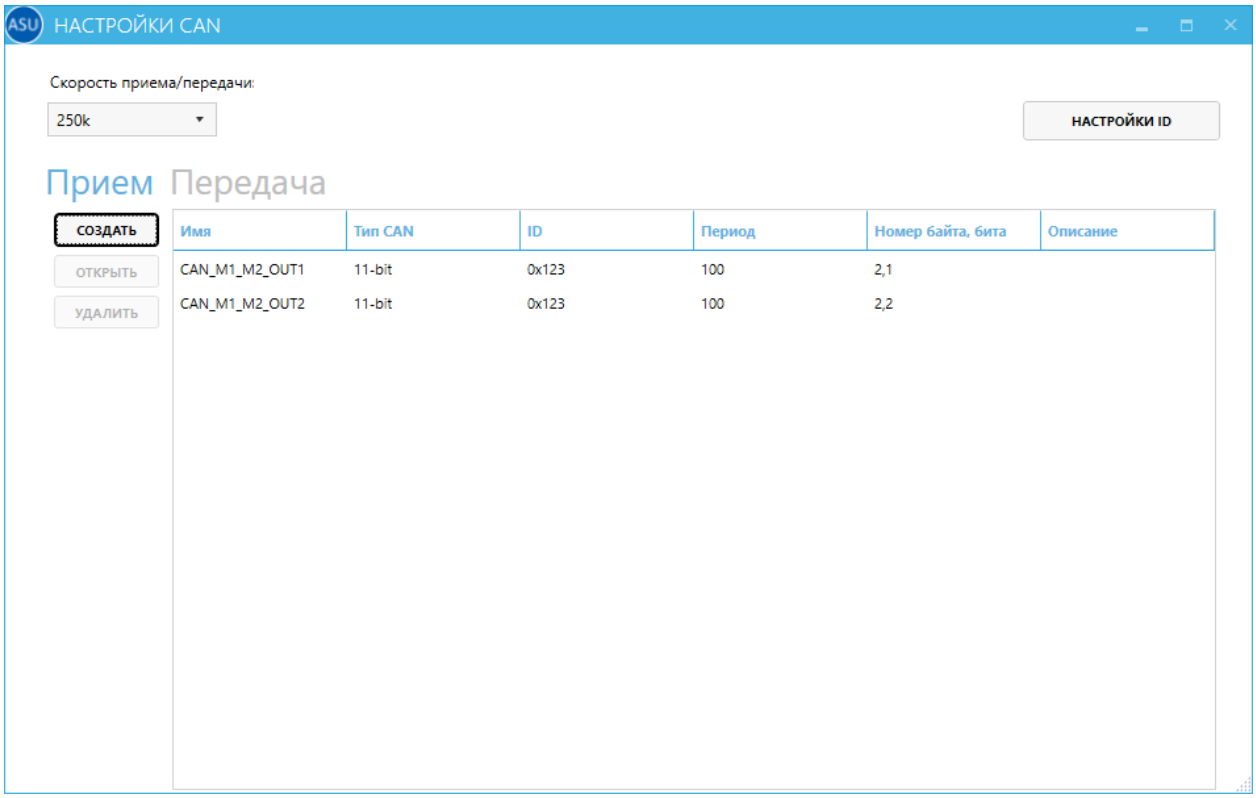
После того, как алгоритм был настроен, перейдем к компиляции проекта и загрузки его на модуль. Перед загрузкой не забудьте изменить ID модуля. По умолчанию все модули имеют ID=5. В случае нахождения в одной сети нескольких модулей с одинаковым ID и попытке прошивки одного из них без отключения от сети, могут возникнуть конфликты (в лучшем случае прошьются все модули с одинаковым ID, в худшем-высветится ошибка загрузки).

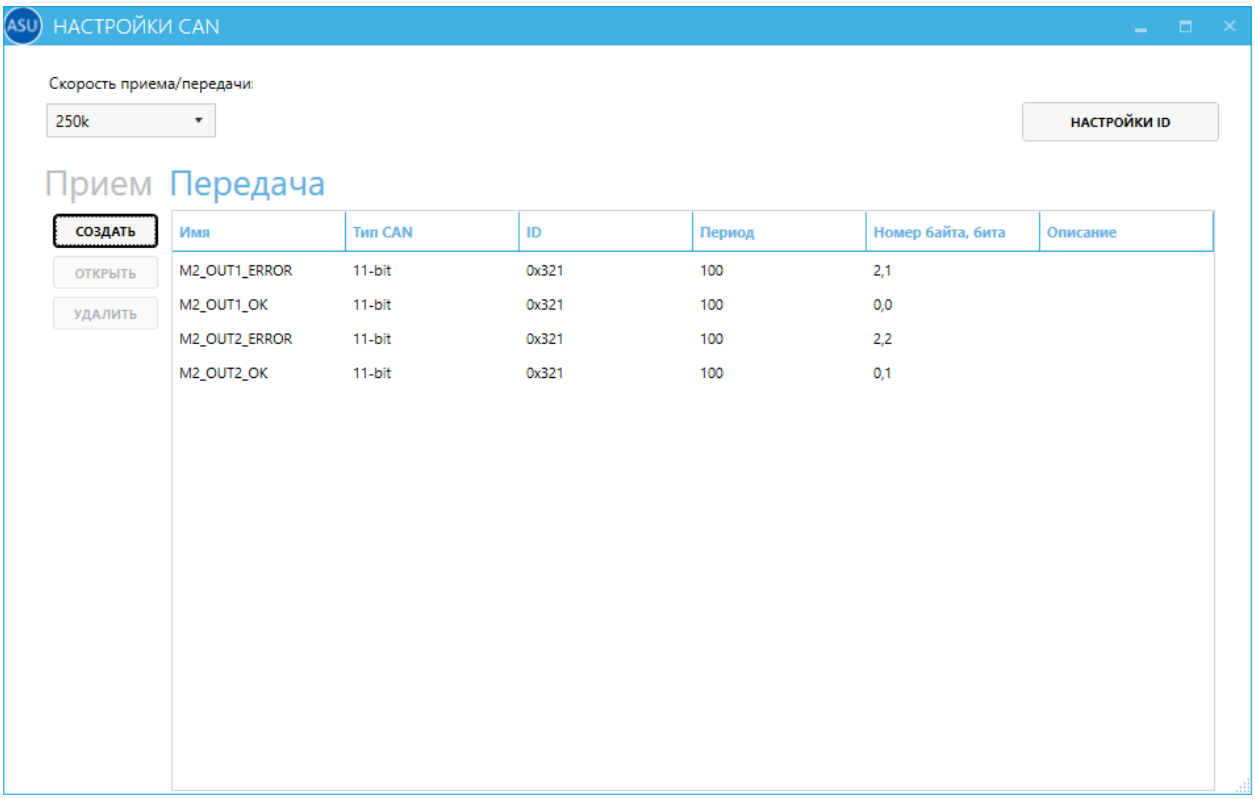



После настройки основного модуля, перейдем к модулям M2 и M3. Первым делом создадим для них соответствующие проекты.



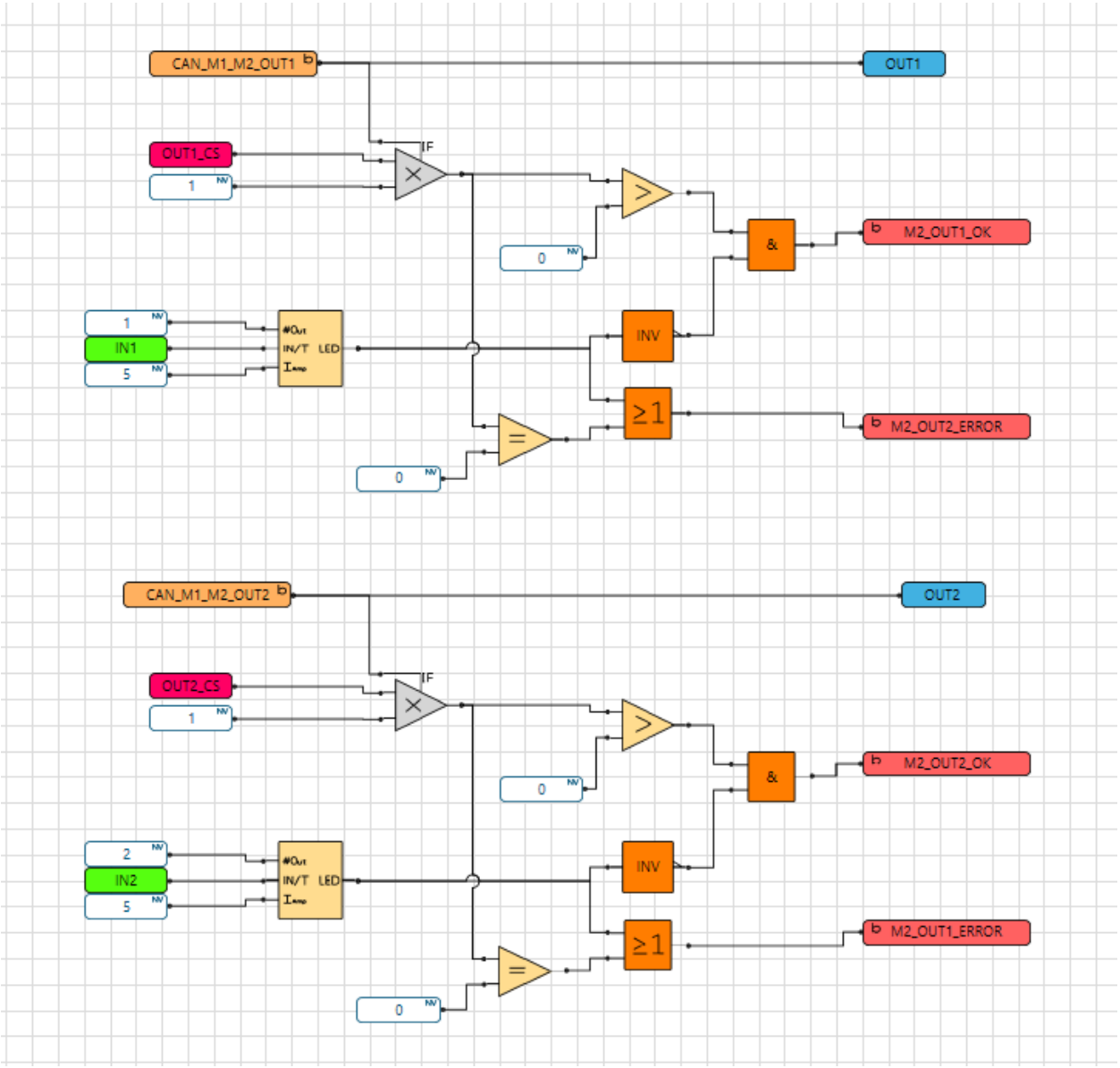

Перейдем к проекту для модуля M2 и создадим CAN-сообщения на прием и передачу, соответствующие сообщениям модуля M1.





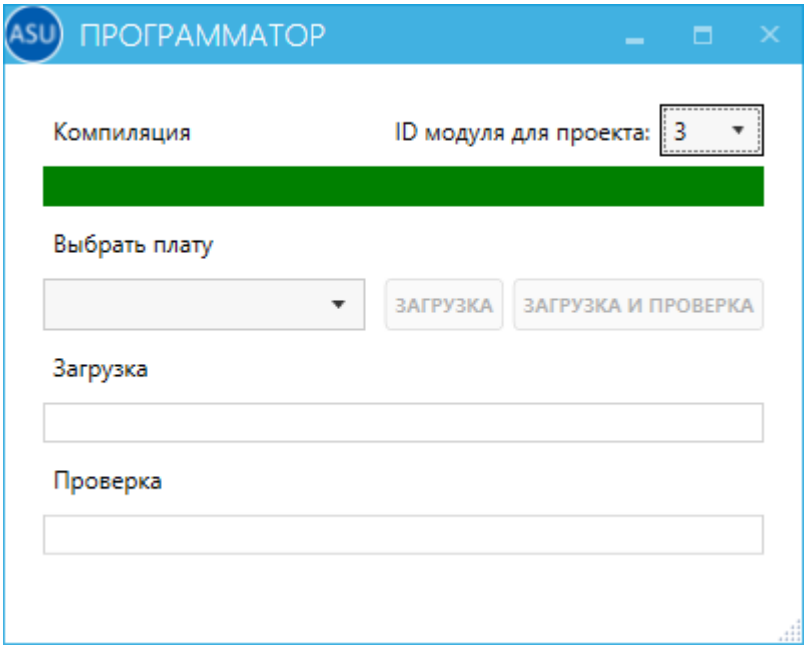

Имена сообщений могут быть любыми, главное, чтобы совпадали ID и номера байтов/битов.

Зададим алгоритм работы модуля М2.

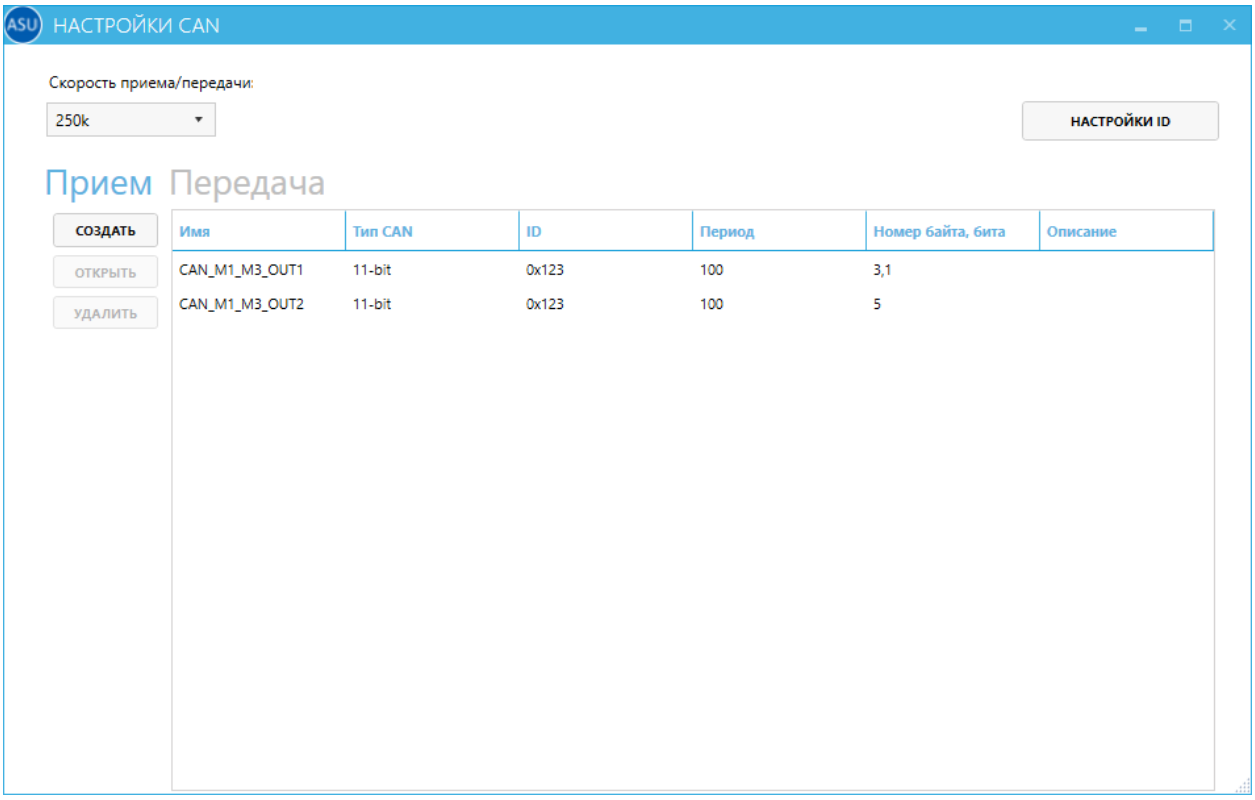


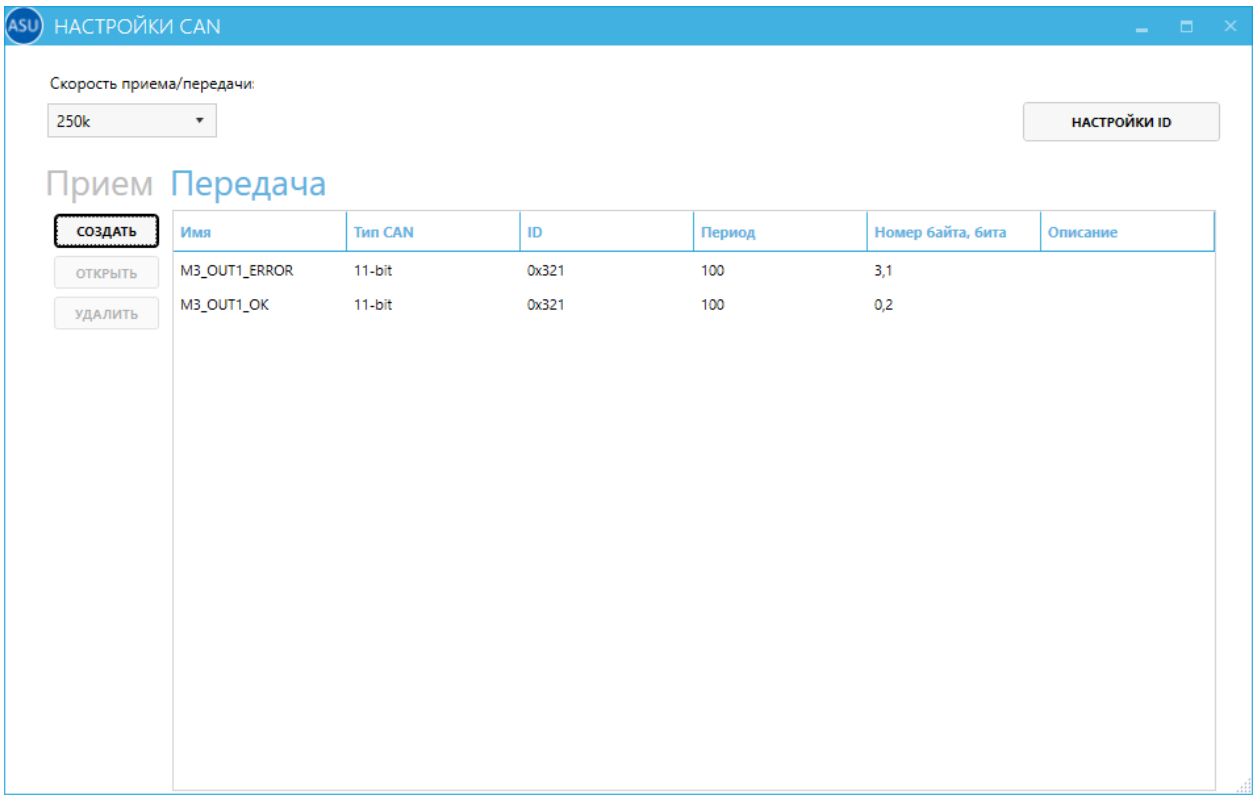

Согласно данному алгоритму, при приходе сообщения от модуля М1 будет активирован одни из выходов (OUT1 или OUT2). В то же время начнется проверка на наличие выходного тока на данном выходе (если ток присутствует, значит есть нагрузка, а значит нет обрыва цепи) и проверка на сработанный предохранитель (КЗ или перегрузка), который в данном случае для обоих выходов назначен на 5А. В случае если оба фактора положительны, в сеть отправляется сообщение, что выход работает. Если же нагрузка на выходе будет по-прежнему соответствовать нулевому значению или сработает предохранитель, в сеть поступит сообщение об ошибке.

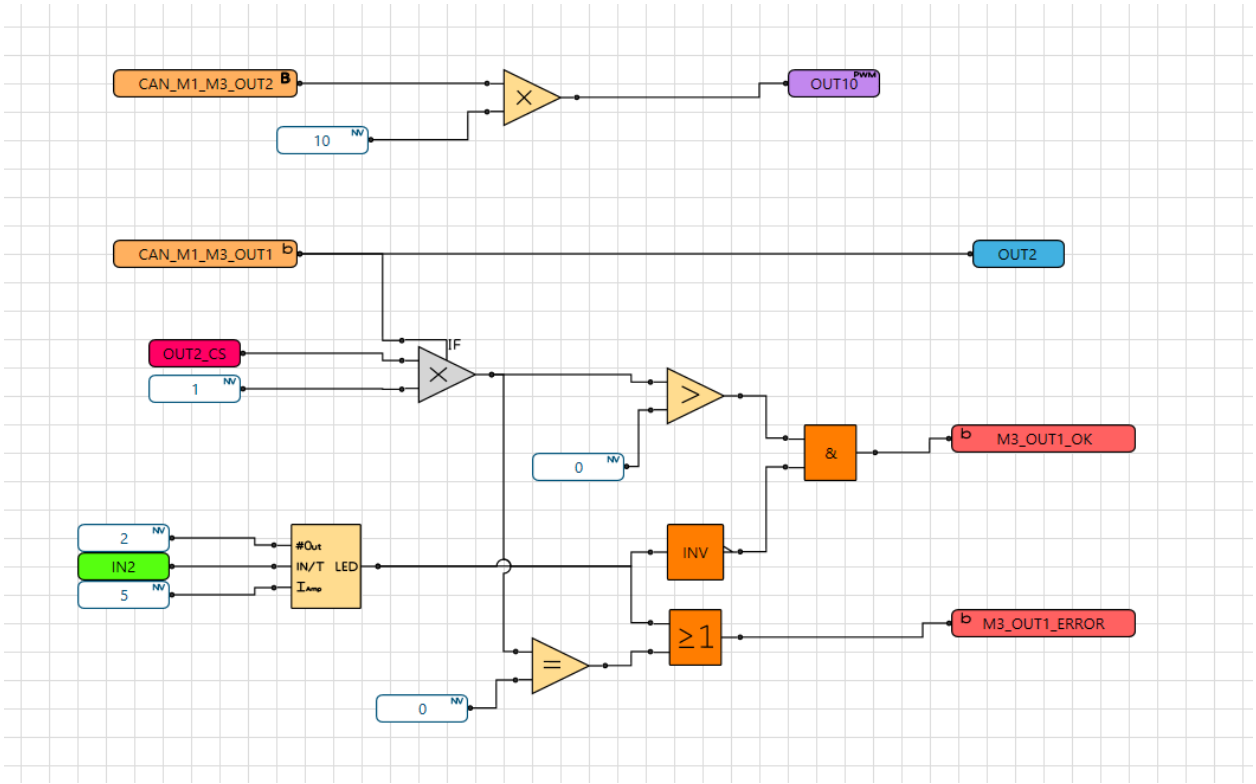
Не забываем сменить ID модуля при загрузке проекта




Аналогично настраиваем проект для модуля М3.






Единственным существенным отличием алгоритма модуля М3 является передача на ШИМ-выход OUT10, значения полученного в байт-сообщении

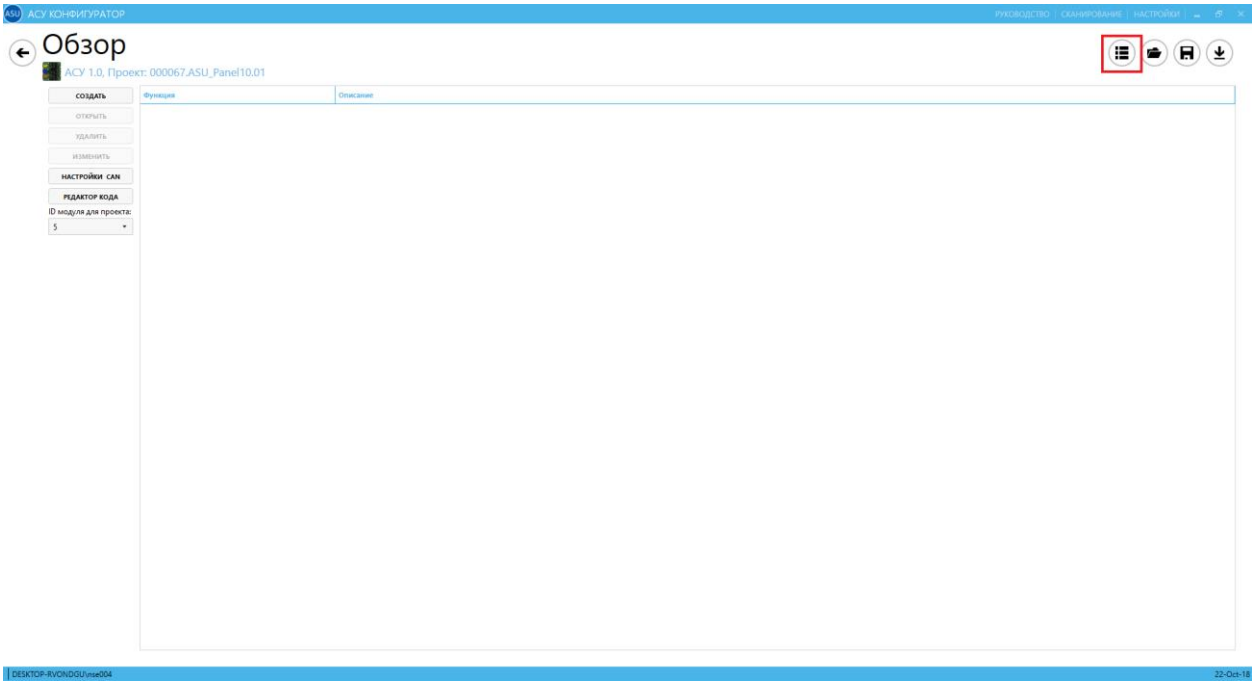
CAN_M1_M3_OUT2 и умноженного на 10. Таким образом при наличии напряжения на AIN5 модуля M1 в диапазоне 0…10В, на выходе ШИМ OUT10 модуля M3 будет присутствовать сигнал с коэффициентом заполнения 0…100%.

# Особенности модулей

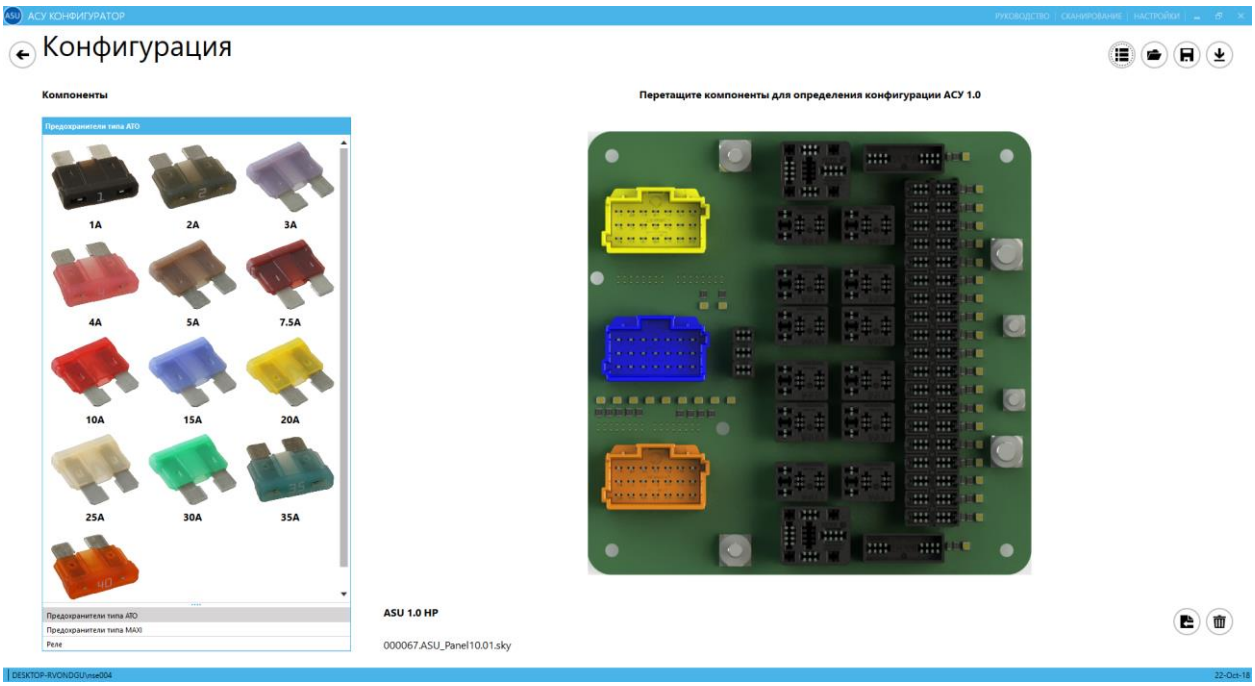
Все примеры руководства имеют общий формат, построенный на базе модуля АСУ 2.1, однако, помимо разных конфигураций входов и выходов, некоторые модули имеют и другие особенности.

## АСУ 1.0

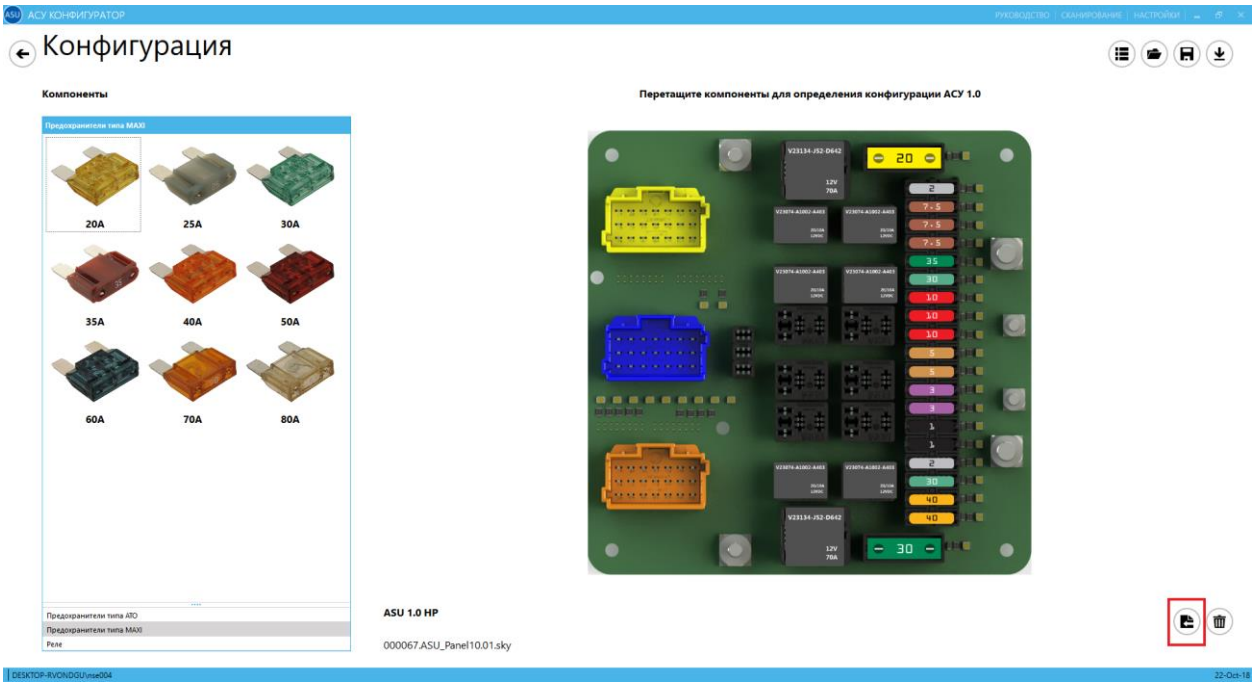
После создания проекта, в окне создания функций пользователю доступна дополнительная кнопка «Показать конфигурацию платы»

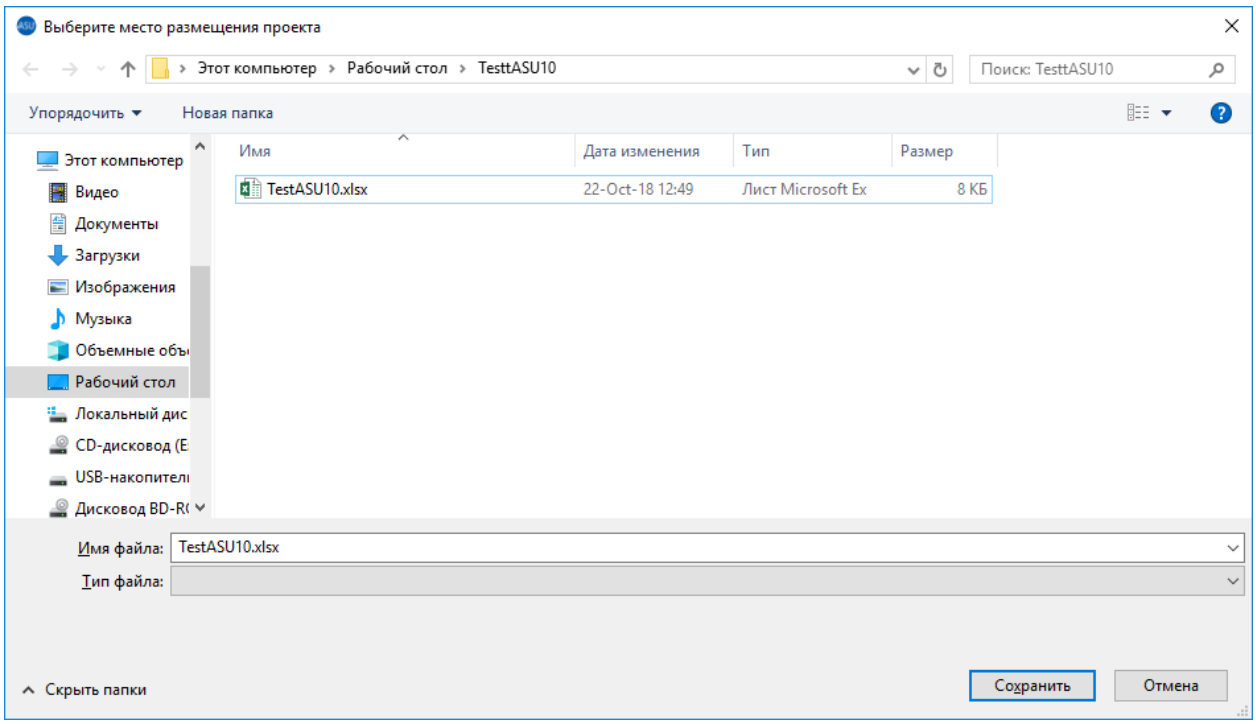



При ее нажатии отображается окно, в котором пользователь может настроить конфигурацию платы и сохранить данные в документе Exel для последующей передачи компании-поставщику.

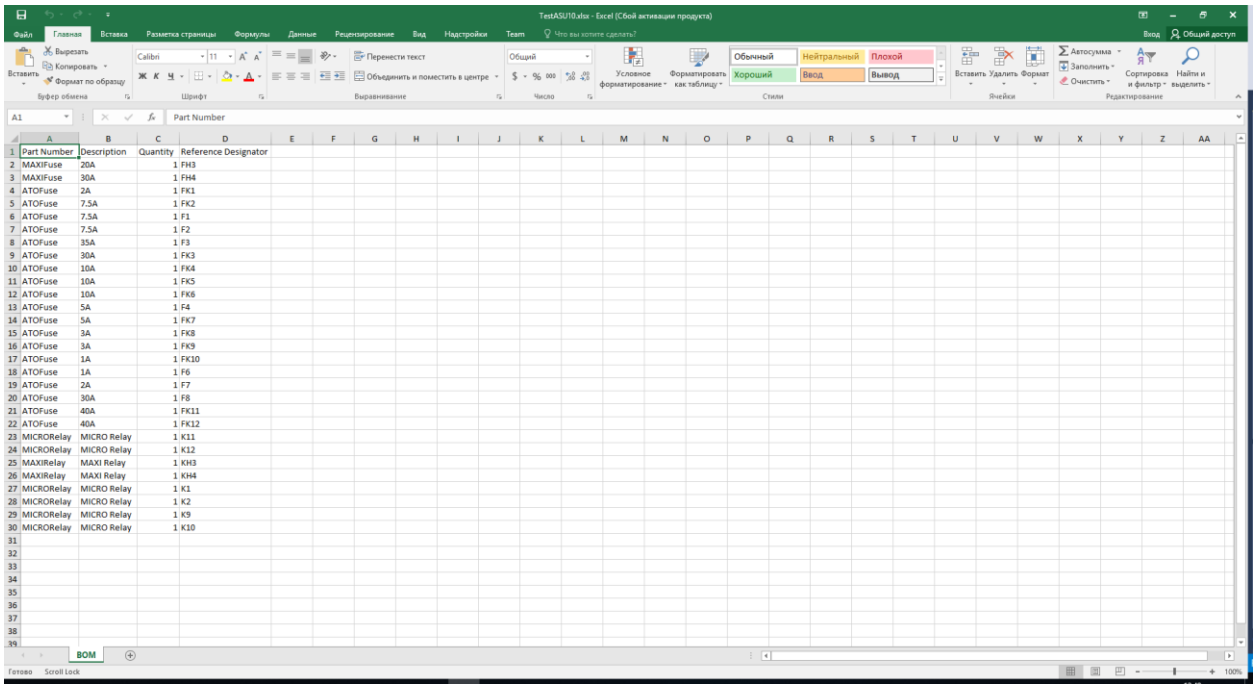



Конфигурация создается путем перетаскивания компонентов из области в левой части экрана на плату. Для экспорта в файл Exel необходимо нажать соответствующую кнопку в правом нижнем углу и выбрать путь сохранения.






После сохранения автоматически откроется файл с перечнем элементов.

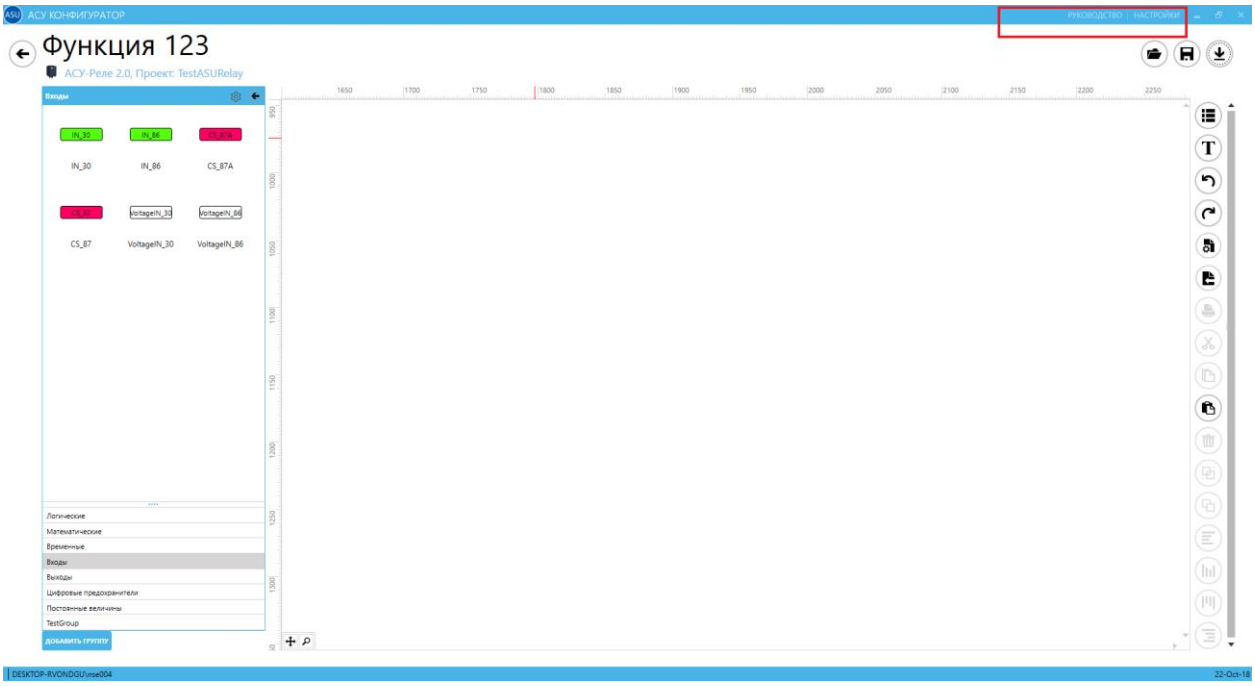


| Part Number | Description | Quantity | Reference Designator |
|---|---|---|---|
| MAXIFuse | 20A | 1 | FH3 |
| MAXIFuse | 30A | 1 | FH4 |
| ATOFuse | 2A | 1 | FK1 |
| ATOFuse | 7.5A | 1 | FK2 |
| ATOFuse | 7.5A | 1 | F1 |
| ATOFuse | 7.5A | 1 | F2 |
| ATOFuse | 35A | 1 | F3 |
| ATOFuse | 30A | 1 | FK3 |
| ATOFuse | 10A | 1 | FK4 |
| ATOFuse | 10A | 1 | FK5 |
| ATOFuse | 10A | 1 | FK6 |
| ATOFuse | 5A | 1 | F4 |
| ATOFuse | 5A | 1 | FK7 |
| ATOFuse | 3A | 1 | FK8 |
| ATOFuse | 3A | 1 | FK9 |
| ATOFuse | 1A | 1 | FK10 |
| ATOFuse | 1A | 1 | F6 |
| ATOFuse | 2A | 1 | F7 |
| ATOFuse | 30A | 1 | F8 |
| ATOFuse | 40A | 1 | FK11 |
| ATOFuse | 40A | 1 | FK12 |
| MICRORelay | MICRO Relay | 1 | K11 |
| MICRORelay | MICRO Relay | 1 | K12 |
| MAXIRelay | MAXI Relay | 1 | KH3 |
| MAXIRelay | MAXI Relay | 1 | KH4 |
| MICRORelay | MICRO Relay | 1 | K1 |
| MICRORelay | MICRO Relay | 1 | K2 |
| MICRORelay | MICRO Relay | 1 | K9 |
| MICRORelay | MICRO Relay | 1 | K10 |

Созданная конфигурация никак не влияет на процесс программирования модуля и призвана лишь упростить процесс их компоновки при заказе.

## АСУ-Реле

АСУ-Реле строятся по отличной от всех остальных модулей технологии и их программирование происходит через отдельный программатор со специальной колодкой. При программировании реле в АСУ Конфигураторе отсутствует раздел «Сканирование». Также, в связи с тем, что реле базируются на микроконтроллерах другой серии, при загрузке проекта необходима предварительная очистка памяти контроллера, из-за чего загрузка кода имеет большую длительность, чем та же процедура в остальных модулях.



# Распространенные ошибки

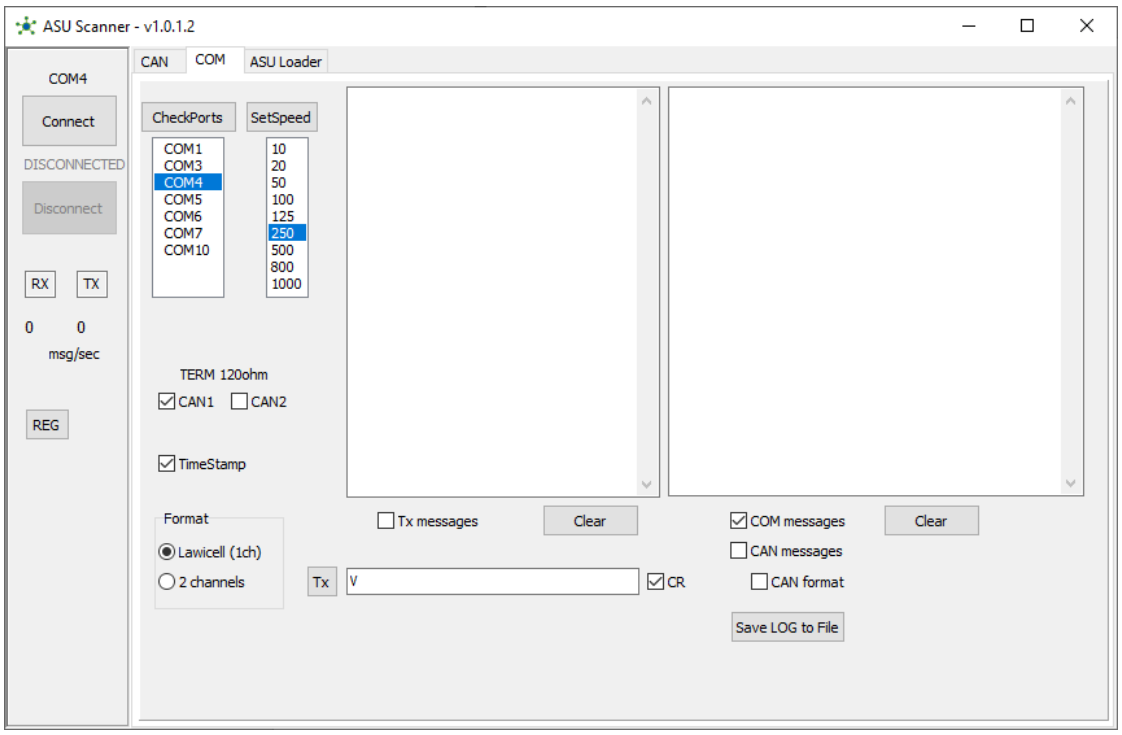
## Не удается найти модуль при сканировании

Проверьте правильность подключения модуля, а также настройки скорости приема/передачи и установку терминатора, попробуйте отключить питание модуля и кабель USB программатора. Проверьте правильность подключения модулей к шине CAN, при подключении нескольких модулей в одну шину может потребоваться установка дополнительного резистора 120 Ом.


Если описанные выше действия не решили проблему, удостоверьтесь, что на шине не висят модули с одинаковым ID, т.к. в таком случае они будут отображаться как один модуль.

Если модуль по-прежнему не виден в системе, попробуйте выполнить запуск его загрузчика вручную. Для этого необходимо выполнить следующие действия

1. Скачать и запустить АСУ сканер
2. Выполнить подключение модуля к программатору, выбрать в АСУ сканере соответствующий COM-port, скорость и нажать Connect



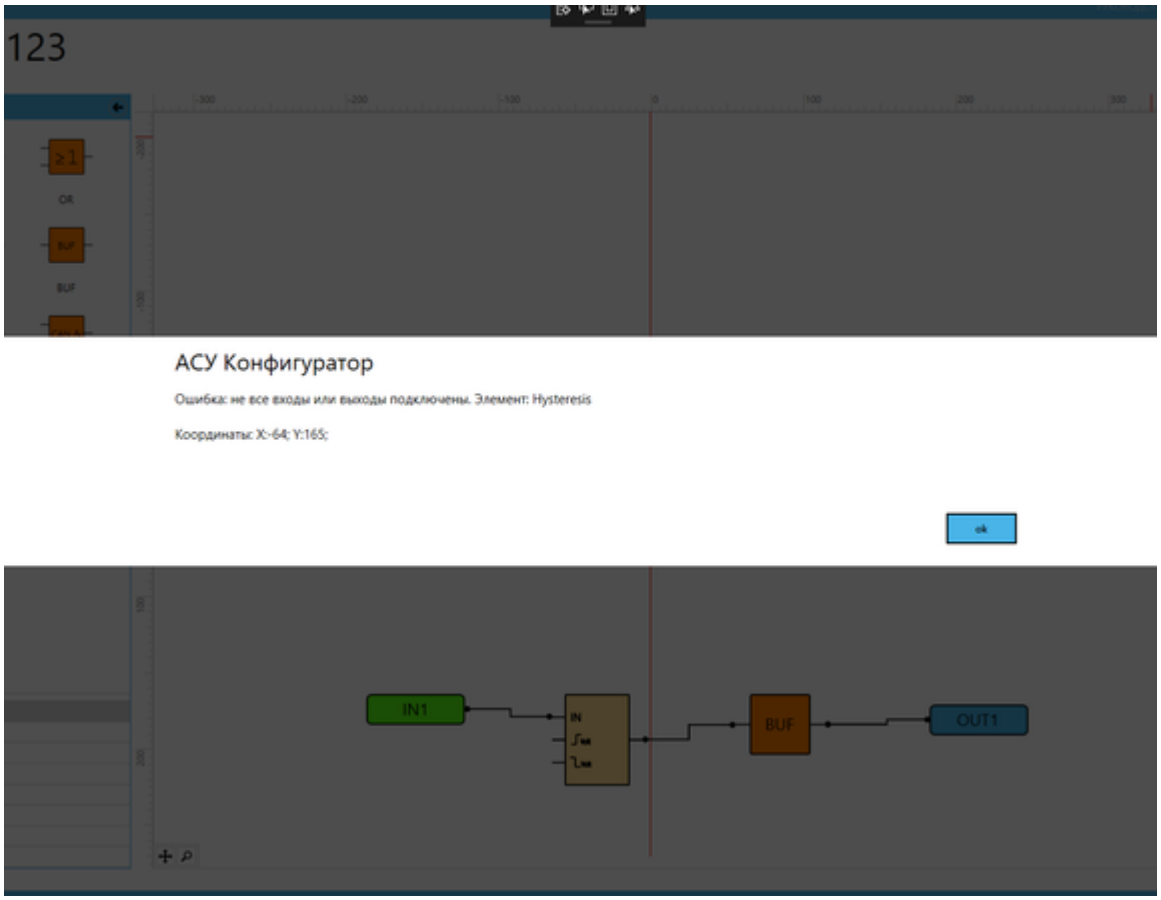

3. Перейти во вкладку ASU Loader, выставить ID модуля (по умолчанию 5, но он мог измениться в соответствии с ID, заданным в проекте), нажать кнопку Lock, после чего выполнить отключение и включение питания модуля, не отключая модуль от программатора.

4. О запуске загрузчика будет свидетельствовать LED1, он должен загореться и не гаснуть. Если вместо этого начали мигать LED1-7, значит ID модуля был выставлен неверно и следует повторить попытку с другим ID (нажимаем StopLock, меняем значение ID, нажимаем Lock, отключаем и подключаем питание модуля).

5. После принудительного запуска загрузчика модуль снова будет виден в АСУ конфигураторе (не забудьте закрыть или выполнить отключение в АСУ Сканере, иначе COM-порт будет недоступен). Принудительный запуск загрузчика может потребоваться, например, если вы прошили модуль АСУ 2.1. прошивкой для АСУ 2.0, не сменив модуль в настройках проекта.

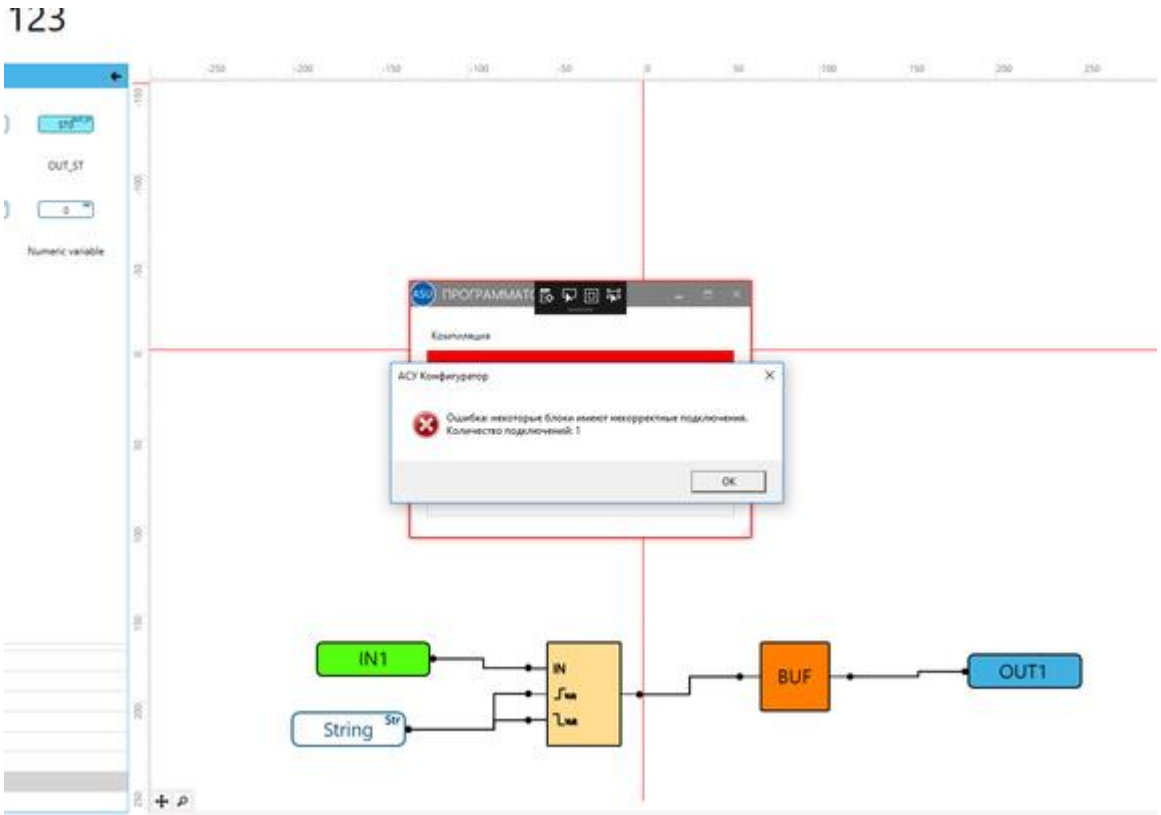
### Блок имеет неподключенные входы/выходы

При создании функции необходимо удостовериться, что все входы и выходы каждого компонента имеют подключения. При обнаружении неподключенных входов/выходов процесс компиляции не запустится и высветится сообщение с названием неподключенного блока и его координатами.



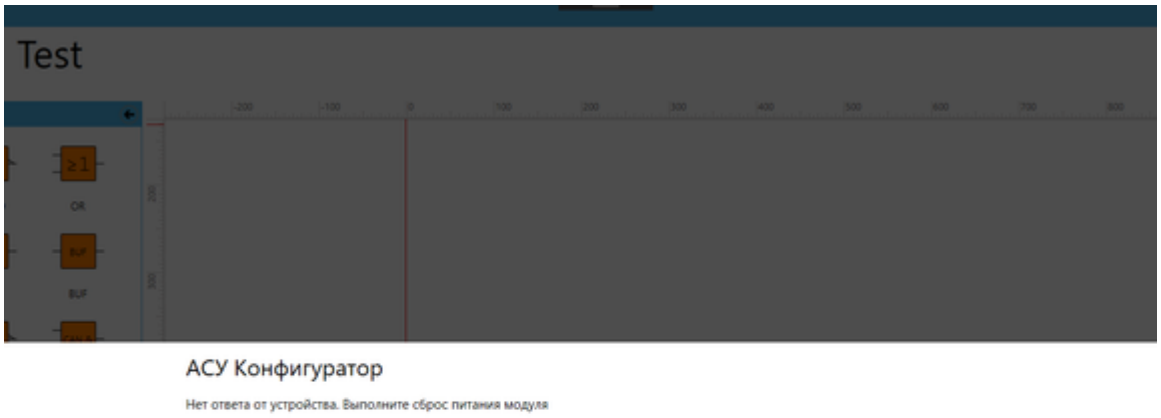
### Блок имеет некорректные подключения

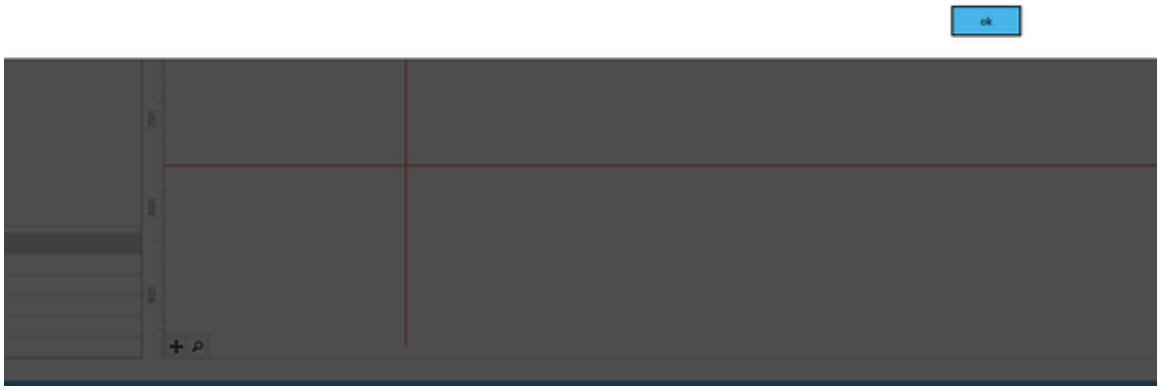
При создании функции необходимо удостовериться, что все входы и выходы имеют корректные подключения: подключения должны соответствовать по типу и должны быть логически обоснованы. Так, при подключении строковой переменной на входы математической функции, процесс компиляции остановится с ошибкой. Стоит также учесть, что некоторые подключения могут проходить компиляцию, но логика их работы будет нарушена, либо будет не соответствовать желаниям пользователя (например, подключение входа измерения напряжения к логическим функциям).



### Нет ответа от устройства

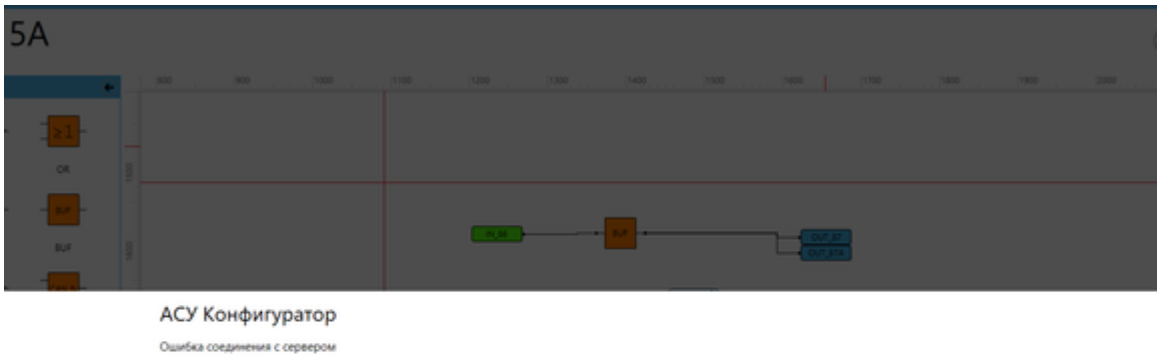
При появлении данной ошибки необходимо проверить правильность подключения устройства, корректность использования данного устройства в текущем проекте (например, проект создан для АСУ реле 1.0, но используется для АСУ реле 2.0). Для возобновления работы необходимо выполнить сброс питания устройства.

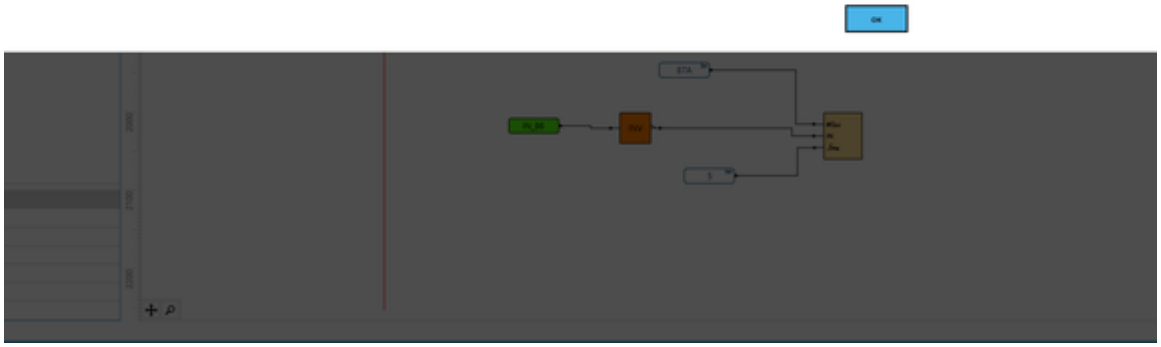


### Ошибка соединения с сервером

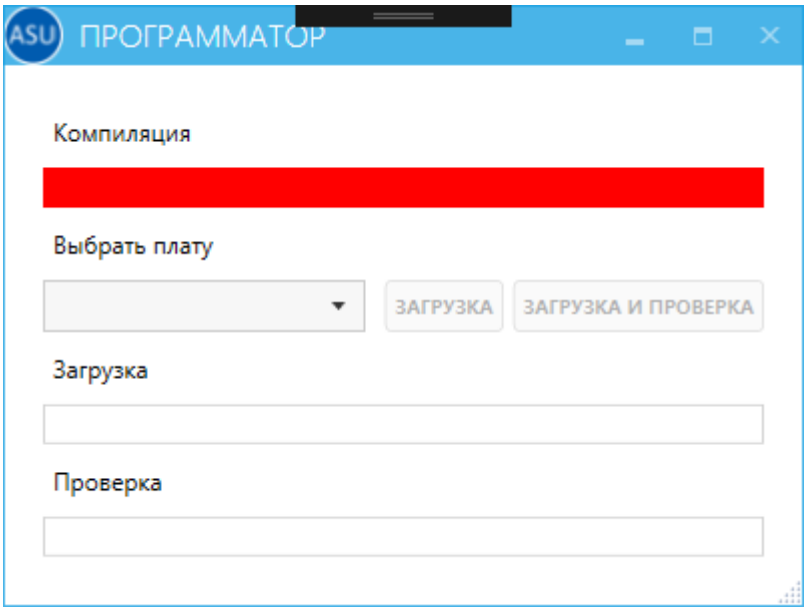
Для компиляции проекта при установленном в настройках пункте «Использовать Интернет-соединение», необходимо подключение к сети Интернет, в случае возникновения ошибки проверьте подключение и повторите попытку.



### Неизвестная ошибка компиляции

Если окно с ошибкой не высветилось, но компиляция не была пройдена - проверьте корректность своих действий с проектом, возможно, вы редактировали проект средствами Windows (переименовывали файл проекта, редактировали содержимое), копировали блоки из старых и неработающих проектов, удаляли какие-либо файлы программы. В случае возникновения подобной ошибки рекомендуется создать новый проект, по возможности не копировать в него блоки из неработающего и связаться со службой поддержки.



# Контакты для связи

В случае возникновения вопросов по функционалу, обнаружении ошибок в программном обеспечении и/или предложениям по нововведениям, просим вас сообщить об этом по приведенным ниже контактам или воспользовавшись кнопкой «Сообщить о проблеме» в разделе настроек.

ООО "Новые Системы Электроники"
ул. Индустриальная, д. 2
214031 Смоленск
Россия

тел. +7-920-322-40-16
E-Mail: info@nse-online.com

Ведущий инженер-программист ООО "Новые Системы Электроники"
Зубарев Святослав
тел. +7-920-330-96-20
E-Mail: svyatoslav.zubarew@nse-online.com