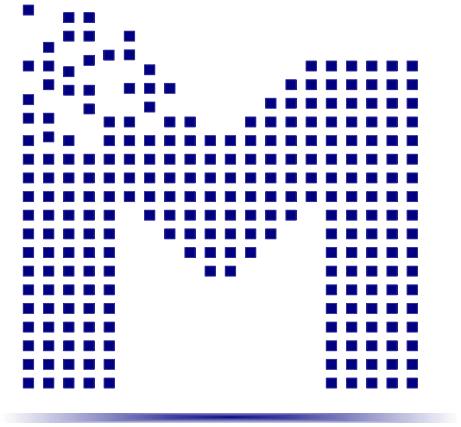


НПФ «Мехатроника-Про»

MexBIOS™ Development Studio

Начало работы – руководство для новых пользователей

Описание интегрированной среды

Краткое содержание

Подробное содержание

Краткое описание системы MexBIOS

Система MexBIOS является программной платформой для создания программного обеспечения микроконтроллеров. Поддерживаются следующие способы разработки программного обеспечения:

- процедурное программирование (написание процедур и функций на языке С)
- программирование функциональными блок-диаграммами
- программирование блок схемами (прорисовывание алгоритмов с учетом ветвлений и последовательностью исполнения формул, в роли которых выступают цепочки функциональных блок-диаграмм)
- автоматное программирование
- событийное программирование.

Процедурное программирование осуществляется путем определения каждого шага в процессе решения задачи, выполняется на языке С в текстовом виде. Особенности используемой версии языка С в MexBIOS зависят от особенностей используемого компилятора, который в свою очередь предопределен программируемым микропроцессором. Большинство современных компиляторов поддерживают компиляцию ассемблерных «вставок», позволяющих ускорить выполнение программы. С помощью процедурного программирования в системе MexBIOS создаются функциональные блоки, которые используется при программировании блок-схемами.

Программирование функциональными блок-диаграммами является способом графического программирования. При программировании используются наборы библиотечных блоков (в том числе и самостоятельно разработанных пользователем). Блоком является подпрограмма, созданная на базе процедурного программирования. Каждый блок имеет входы и выходы для данных. Пользователь выбирает необходимые блоки и соединяет входы и выходы блоков в соответствии с решаемой задачей.

Программирование блок-схемами является способом графического программирования. При данном подходе пользователь конструирует граф-схему алгоритма, в которой отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими исполнение последовательности. При программировании в среде MexBIOS используются следующие блоки: начало-конец алгоритма, блок действий, логический блок (блок организации условного ветвления), блок цикла.

Автоматное программирование в среде MexBIOS является способом графического программирования и осуществляется путем задания основных состояний управляемой системы, действий, характерных для каждого состояния, условий смены состояний.

Событийное программирование осуществляется путем формирование событий, которые запускают последовательности действий, формализованные любым из предыдущих способов. В качестве событий, возможно использовать как события, возникающие на аппаратном уровне, так и при изменении значений в памяти данных.

Принцип действия системы MexBIOS

В микроконтроллере пользователем периодически вызывается ядро системы. Ядро системы вместе с библиотекой программных блоков должно быть предварительно загружено в память микроконтроллера. Ядро выполняет запуск программных блоков в соответствии с информацией (байт-кодом), предоставленной пользователем. Программные блоки компилируются стандартными средствами компиляции и загружаются пользователем перед запуском ядра. Байт-код формируется пользователем путем графического программирования, в котором указываются события и условия запуска алгоритмов, алгоритмы в виде блок схем, функциональные блок-диаграммы, а также порядок их исполнения и адреса считывания/записи данных программных блоков. В результате исполнения программных блоков в соответствии с пользовательскими алгоритмами, происходит изменения данных в памяти микроконтроллера в функции от поступающих данных и возникающих событиях. В дальнейшем понятие «байт код», «конфигурационный файл», «матрица» равнозначны.

Примечание: в виду того, что исполняемые блоки являются предварительно откомпилированными, а ядро динамически формирует последовательность и условия их исполнения, производительность системы MexBIOS практически соответствует производительности откомпилированного кода.

Система **MexBIOS** состоит из следующих **основных частей:**

- интегрированная среда MexBIOS Development Studio
- библиотека блоков
- ядро
- «стартовый» проект
- набор готовых шаблонов для проектирования.

Интегрированная среда (далее ИС) позволяет проводить разработку программы для микроконтроллера и ее отладку.

ИС поддерживает способы проектирования:
Создание программных блоков текстовым способом
Конструирование формул из функциональных блок-диаграмм
Конструирование алгоритмов
Конструирование машины состояний
Задание событий с указанием источника их возникновения и алгоритмов обработки

ИС поддерживает способы отладки программ:
Редактирование значений переменных
Вывод значений наблюдаемых переменных
Вывод значений наблюдаемого сигнала в виде массива
Предварительное моделирование работы кода совместно с моделями объектов управления

ИС позволяет также выполнять:
Создание интерфейсов управления с помощью графических компонентов.
Проводить обмен данными с внешними приложениями по стандартным протоколам передачи данных.

Существует два режима работы ИС: моделирования и обмен данными с микроконтроллером. Режим моделирования предназначен для имитации работы созданной программы на компьютере. Для режима моделирования создана специальная библиотека Models, которая предназначена для создания математических моделей объектов регулирования. Режим обмена данными с чипом предназначен для работы с аппаратным обеспечением, в которое загружено ядро «MexBIOS™ Kernel». Ядро представляет собой библиотеку блоков (для каждой серии процессоров своя библиотека) и ядро операционной среды реального времени «MexBIOS». По созданной и проверенной в режиме моделирования программе создаётся конфигурационный файл, который загружается в память чипа. После загрузки конфигурационного файла в память чипа, «MexBIOS™ Kernel» по загруженным данным создаёт структуру идентичную визуальной программе собранной в MexBIOS™ Development Studio. Работа программ идентична, после установки связи с чипом можно произвести считывание и задание параметров в чипе.

Стартовый проект осуществляет запуск системы MexBIOS в микроконтроллере. Стартовый проект обеспечивает привязку системы к микроконтроллеру и схемотехническим решениям электронной платы, на которой установлен микроконтроллер. Стартовый проект зависит от типа используемого микроконтроллера. Стартовый проект содержит вызов ядра системы MexBIOS и обеспечивает коммуникации с ИС. Производителем прилагается пример стартового проекта к системе для поддерживаемых типов микроконтроллеров. Ядро системы MexBIOS может получать данные из стартового проекта и передавать данные в стартовый проект посредством специальных блоков. В качестве стартового проекта, пользователь может использовать собственный проект, включив в него ядро MexBIOS™ и библиотеку блоков.

Ядро осуществляет выполнение программных блоков в соответствии с порядком и условиями запуска, предопределенными пользователями путем графического программирования. По сути, является интерпретатором байт-кода. За счет того, что ядро запускает предварительно откомпилированные процедуры, общая производительность системы сравнима с проектами, созданными путем полной компиляции.

Библиотеки блоков создаются путем текстового программирования с последующей компиляцией. Библиотека блоков загружается в память микроконтроллера, параллельно с этим формируется аналогичная библиотека блоков для моделирования работы системы на персональном компьютере. Библиотеки содержат стандартные группы блоков, которые наиболее часто используются при проектировании программного обеспечения для микроконтроллеров, в частности, для систем управления электродвигателями. Создание блоков и формирование библиотек осуществляется в ИС.

Набор готовых шаблонов представляет собой пакет готовых проектов прикладного программного обеспечения для системы MexBIOS™, значительно упрощающих разработку программного обеспечения.

Технология использования среды MexBIOS

1. Пользователь встраивает вызов ядра системы MexBIOS в существующий проект программного обеспечения либо использует предоставленный в составе системы пример стартового проекта, где вызов ядра системы уже встроен. Вызов ядра устанавливается в процедуры обработки прерываний, допускается вызов ядра из нескольких процедур обработки прерываний.
2. Загрузочный файл стартового проекта загружается в память микроконтроллера в установленные сектора памяти.

3. Пользователь при необходимости дополняет стандартную библиотеку блоков собственными программными блоками через ИС. В результате получается загрузочный файл с откомпилированными программными процедурами и ядром системы.
4. Библиотека блоков загружается в память микроконтроллера в сектора памяти, не занятые стартовым проектом.
5. Пользователь назначает события, на основании которые происходит запуск алгоритмов.
6. Пользователь детализирует алгоритмы, устанавливая условия ветвления и вводя формулы, которые обеспечивают исполнение программных блоков, при необходимости настраивает машину состояний.
7. Пользователь проводит предварительное моделирование работы созданного программного обеспечения, для имитации сигналов с объектов управления используются модели объектов управления из стандартной библиотеки моделей либо разработанные пользователем.
8. Пользователь загружает байт-код в микроконтроллер посредством последовательного интерфейса средствами ИС и запускает программу на исполнение.
9. Посредством средств отладки корректируются необходимые координаты, на экран компьютера выводятся осциллограммы сигналов. Для визуализации сигналов и данных и облегчения задания координат пользователь формирует панель управления из стандартных визуальных компонентов.
10. На основании полученных данных принимаются решения о корректировке разработанных алгоритмов.

Ключевые правила для разработки программного обеспечения в ИС

1. Переменные объявляются только в корневом поле набора. Необходимо следить за тем, чтобы переменные объявлялись в «правильном» поле набора. Существует 2 корневых поля набора - для Моделей и для микроконтроллера.
2. Прямой доступ к входам и выходам блоков и к переменным осуществляется через блоки ввода и вывода (соответственно IN и OUT). Блоки также можно соединить между собой функционально с помощью линий связи.
3. Элементы типа State позволяют вычислять только формулы, формируемые из последовательностей функциональных блоков, вычисление алгоритмов внутри State не возможно.
4. Типы данных должны назначаться пользователем в соответствии с решаемыми задачами и применяемыми блоками. Для преобразования форматов данных необходимо использовать стандартные блоки.
5. Блок условного ветвления использует для сравнения переменных. Необходимо задавать тип данных переменных для сравнения, соответствующий типу сравниваемых переменных в настройке блока ветвления.
6. Необходимо задать события, по которым происходит запуск блок схем. Блок-схемы в свою очередь запускают другие блок-схемы и формулы. Формулы (как конечный элемент расчетов) обеспечивают расчет согласно схемам соединений функциональных блоков.
7. Если событие является аппаратным, то оно должно быть соответствующим образом сконфигурировано в стартовом проекте и добавлен вызов ядра системы из процедуры обработки этого события.
8. Передача значений между различными формулами осуществляется либо через переменную, либо (когда создание переменной нецелесообразно) согласно элементам **Разрыв связи (TP_IN, TP_OUT)**
9. При создании формул (цепочки исполнения функциональны блоков) необходимо блоки вывода результата помещать на поле редактирования последними. Порядок исполнения блоков вывода Out можно посмотреть с помощью кнопки Block Calls – его номер должен быть выше, чем всех остальных блоков формулы.

10. При разработке программ в ИС идет использование динамической памяти компьютера, в связи с чем, возможно возникновение конфликтов с компьютерами, имеющими индивидуальные особенности настройки либо низкое быстродействие. В случае конфликтов операционной системы компьютера с системой MexBIOS, проявляющееся в выводе «аварийных» сообщений, необходимо сохранить проект и перезапустить ИС.

Особенности разработки программного обеспечения в ИС

1. Стартовый проект формируется, как правило, в среде программирования, прилагающийся к используемому микроконтроллеру.
2. Возможно подключение внешнего загрузчика программы во флеш-память микроконтроллера, вызываемого из ИС.
3. Допускается загружать (если это позволяет лицензионного соглашение) конфигурационный файл вместе с библиотекой, для чего необходимо установить соответствующие опции в окне настроек компилятора во флеш память.
4. Выполнение программы в микроконтроллере происходит сразу после загрузки матрицы, обновление данных в ИС происходит только после нажатия кнопки **Начать обновление**.
5. Используя значения адресов параметров в окне **Таблица переменных**, можно в последующем по данным адресам обмениваться данными без ИС, с использованием стандартных протоколов.
6. Связь со стартовым проектом (обмен данными) осуществляется через подключение map-файла в окне **Параметры**, вкладка **Домой**.
7. Стартовый проект может содержать прежние проекты пользователя, в нем реализуются алгоритмы, сложные для создания в ИС, либо требующие максимального быстродействия.
8. При закрытии всех панелей редактирования их вызов возможен только через окно Менеджера проекта.
9. При наблюдении за изменением сигналов используется блок виртуального осциллографа (далее просто осциллографа). Для режима моделирования применяется осциллограф в режиме Simple. Для наблюдения за изменениями сигналов в реальном времени на контроллере необходимо применить буферный осциллограф. Более подробную информацию смотрите в соответствующих разделах данного руководства.
10. При запуске обновления в режиме обмена данными с микроконтроллером, одновременно запускается и процесс моделирования. Данный эффект можно использовать для сложных вычислений, не требующих расчетов в реальном времени, с последующей передачей рассчитанных данных в микроконтроллер посредством элемента **Разрыв связи (TP_IN, TP_OUT)**
11. Моделирование можно замедлить, для этого необходимо задать параметр **Замедление моделирования** вкладка **Домой**, кнопка **Параметры**, раздел **Моделирования**.
12. События, по которым запускаются алгоритмы, могут представлять собой, в том числе, и сравнение значений двух переменных. При этом алгоритм обработки таких событий запускается с самым низким приоритетом.

Типы данных

1. Программирование микроконтроллеров неотрывно связано с операциями над числами. В отличие от настольных компьютеров с практически не ограниченным размером числа, в микроконтроллерах применяются числа ограниченные длиной слова. У современных микроконтроллеров длина слова составляет 32 бита.

2. Существенно расширить диапазон вычисления позволяет применение чисел с плавающей запятой, однако их применение требует специального вычислительного модуля в процессоре.

3. С помощью технологии фиксированной запятой и применения относительных величин, возможно решать задачи требующие высокоточных вычислений на микроконтроллерах с целочисленной арифметикой.

Примечание: Процессоры с фиксированной запятой имеют возможность производить математические вычисления в формате Float, однако такие вычисления требуют значительных затрат вычислительной мощности процессора и применяются лишь в случаях, когда необходимо добиться большой точности вычисления.

4. В целочисленных микроконтроллерах арифметические операции проводятся с применением фиксированной запятой. Дробные числа, определённым образом, преобразуются в целые числами. При этом под представление дробной и целой части числа выделяется различное количество бит слова контроллера (число выделенных бит задаёт пользователь, выбирая определённый формат представления).

5. Для большей наглядности приведём пример. В 32х битном микроконтроллере числа могут быть в диапазоне от (-2^{31}) до ($2^{31}-1$). Чтобы представить дробное число с помощью целого числа 31 бит, необходимо под дробную часть числа назначить количество бит (выбирается из необходимой точности представления числа), оставшееся число бит пойдёт на представление целой части. Пусть нужно представить число π в формате 24, то есть под целую часть отведено 7 бит, под дробную 24, тогда:

3.141592653589793238462643383279 5* 2^{24}=52707178,533

6. Так как числа целые – дробная часть отбрасывается. Если перевести обратно в дробное число делением полученного числа на 2^{24} получим число π с точностью равной $1/2^{24}$.

52707178/2^{24} = **3.1415926**21803284

7. Для облегчения работы с дробными числами на целочисленных микроконтроллерах применяется библиотека IQmath. В библиотеку включены функции для работы с числами, представленными в различных форматах. Числом в формате IQN будем называть число, для представление дробной части которого выделено N бит.

8. Знак входит в биты, выделенные под целую часть.

Способы доступа для переменных

Блок **VAR** представляет собой аналог переменной в классическом программировании. Вытаскивая блок на поле набора, вы создаёте переменную, к которой можно обращаться (считывать, записывать) с помощью блоков **IN** и **OUT** в любом месте программы (в пределах библиотеки).

Используя блоки IN, OUT, VAR пользователь получает гибкий инструмент для решения своих задач.

Если ядро MexBIOS™ подключено в проект пользователя, то с помощью блоков RD_MEM и WR_MEM можно осуществить чтение и запись в переменные проекта пользователя. Для этого необходимо подключить Map-файл, генерируемый, при создании проекта пользователя в окне **Параметры**. Подробней смотрите раздел «**Обращение к переменным в стартовом проекте**».

Основные компоненты ИС

1. Поле набора.
2. Библиотека компонентов.
    a. Встроенные компоненты.
    b. Элементы графического управления.
    c. Блоки моделей.
    d. Библиотека блоков.
    e. Протоколы данных.
3. Модуль создания блоков.
4. Инспектор свойств блоков.
5. Панель имитации работы системы прерываний микроконтроллера.
6. Менеджер проекта. Функция поиска блоков.
7. Окно настройки режима моделирования.
8. Окно настройки компиляции библиотеки.
9. Окно просмотра/задания значений переменных.
10. Окно просмотра параметров проекта.

Система лицензирования операционной среды MexBIOS Kernel

Среда MexBIOS Development Studio оснащена системой лицензирования, основанной на аппаратных ключах HASP. **HASP** (*Hardware Against Software Piracy*) – это мультиплатформенная аппаратно-программная система защиты программ и данных от нелегального использования и несанкционированного распространения.

Для работы среды в полнофункциональном режиме необходимо в USB-порт компьютера подключить электронный ключ HASP с лицензией. При первой установке ключа операционная система может потребовать установить драйвер.

Файл установленной программы находится по адресу **C:\Users\%USERNAME%\AppData\Roaming\NPF Mechatronica-Pro\MexBIOS Development Studio**.

Демонстрационный режим. Без лицензии

Доступны все возможности системы по составлению схем из имеющихся в палитре блоков, загрузки и отладки программы в ОЗУ (RAM), время работы среды и системы исполнения задач в контроллере неограниченно.

Ограничения Демонстрационной версии:

- Отсутствует возможность сохранять в энергонезависимую память конфигурационный файл.
- Допускается добавление не более 15 блоков пользователя.
- Ограничение в 1024 символов (не считая пробелы) на создание блока пользователя.
- Ограничение на объем кода генерируемый функцией генерации кода.

Примечание: Для демонстрационного режима доступна загрузка только в RAM. Время работы проекта не ограничено, пока есть питание на процессоре.

В данном режиме собирается и отправляется на сервер производителя ИС следующая информация: время запуска, количество используемых блоков, время завершения сеанса работы. Отключить отправку данной статистической информации возможно через окно настроек **Параметры**.

Статистическая информация используется производителем исключительно для оценки масштаба и периодичности применения системы.

В режимах, отличных от демонстрационного, статистическая информация не отсылается, однако переход в другой режим требует покупки лицензии.

Режим лицензирования

Ограниченная лицензия

Предоставляется через покупку HASP-ключа, который разрешает сохранять файл программы пользователя в энергонезависимую память либо вставлять его в библиотеку функций системы.

Стандартная лицензия

Предоставляется через покупку HASP-ключа, в который внесена информация о снятии ограничений на количество блоков пользователя библиотеки и размера функций библиотеки.

Неограниченная лицензия

Предоставляется через покупку HASP ключа, в который занесена информация о снятии всех ограничений на систему (в том числе по размеру генерируемого текстового кода).

Режим лицензирования снимает все временные ограничения на работу ядра в составе микроконтроллера. Для включения режима лицензирования необходимо:
1. Установить в USB-порт компьютера аппаратный ключ HASP.
2. После включения программы над полосой загрузки отобразится дополнительная информация о лицензии.

Режим лицензирования контролирует количество загрузок программ в контроллер. Также возможно существование ограничения по количеству внешних блоков библиотеки чипа, добавленных на схему.

Количество доступных загрузок программ в контроллер определяется лицензионным соглашением и отображается в памяти ключа. По окончании загрузки программы в контроллер количество доступных загрузок программ уменьшается на одну.

Приобретая ключ, вы соглашаетесь с условиями **лицензионного соглашения**.

Содержание папки программы

Программа устанавливается по следующему адресу:

C:\Users\%USERNAME%\AppData\Roaming\NPF Mechatronica-Pro\MexBIOS Development Studio

Вместе с программой устанавливаются следующие папки:

Help – папка содержит *.pdf файлы руководства по программе.

Libraries\[Lib name]\Examples – папка содержит файлы *.mbp и *.pdf примеры и описание примеров применения. Примеры, содержащие в названии um (сокращение от User Manual) описаны в данном руководстве. Примеры, содержащие в названии ex (сокращение от examples) представляют собой примеры работы групп блоков в библиотеке.

Config – папка содержит различные файлы **mbcfg** конфигурации памяти.

Video lessons – папка содержит видео уроки по применению системы.

Расширения файлов MexBIOS Development Studio

***.mbp** – файл проекта программы MexBIOS Development Studio.
***.mbd** – конфигурационный файл, загружаемый в память контроллера.
***.out** – откомпилированный файл библиотеки или стартового проекта пользователя.
***.hex** – файл, который получен из файла *.out, для последующей загрузке по SCI.
***.mbcfg** – файл конфигурации распределения памяти.
***.map** – файл с информацией об адресах переменных стартового проекта. Генерируется компилятором C2000.
***.mbdat** – файл с коэффициентами для блока либо для вкладки коэффициенты.

Термины и определения

Ядро – программный интерпретатор. Основное отличие ядра от интерпретаторов в широком смысле этого слова, состоит в том, что для исполнения программ ему необходимо наличие предустановленных библиотек блоков. Ядро формирует порядок и условия запуска блоков в соответствии с информацией, полученной из графической программной среды MexBIOS™ Development Studio и представляется в виде матрицы (байт-кода), а также потоки данных.

Библиотека блоков – это откомпилированный исполняемый модуль. Содержит в себе откомпилированные процедуры блоков, которые могут быть задействованы при наборе графической программы в среде MexBIOS™ Development Studio. Используется ядром, при исполнении графической программы представленной в матрице, путём вызова процедур соответствующих блокам присутствующих в графической программе.

Матрица – информация о порядке и условиях запуска блоков, а также потоков данных в графически разработанной программе. По сути представляет байт-код.

Связь – элемент графического языка, структурная единица графической программы, предназначенная для представления потоков данных между блоками и очерёдности их вызова.

Схема – совокупность блоков, представленных на одном поле набора. Различают три вида схем: ALGORITHM, STATE_FLOW и FORMULA.

Поле набора – поле для визуального представления блоков, участвующих в графической программе.

Главное поле набора – поле набора для создания структуры программы, начало программы.

Корневое поле набора – верхний уровень структуры программы.

Корень Алгоритма – вложенный в алгоритм уровень поля набора структуры программы.

Поле набора – Поле набора внутри блоков FORMULA и STATE.

Блок – по отношению к среде различают два вида блоков: встроенные (Embedded) и внешние блоки. В палитре блок представляется графическим элементом, имеющий характерные внешние особенности и может быть вынесен из палитры на наборное поле для составления графической программы.

Встроенный блок – элемент графического языка, предназначенный для представления таких конструкций графического языка как событие, формула, алгоритм, цикл, ветвление, state-машины, разрыв связи (телепорт).

Внешний блок – структурная единица библиотеки блоков. Представляет собой функционально обособленный элемент, для выполнения заложенной в него функции. Далее, если не упомянуто особо, просто блок.

Функция внешнего блока – последовательность действий, объединённая общим функциональным назначением блока. Представляется в виде исходного кода на языке С. Представляет собой откомпилированную функцию, которая является частью библиотеки блоков и вызывается непосредственно ядром при исполнении графической программы, представленной в виде матрицы.

Событие (EVENT) – элемент, предназначенный для представления в программе событий аппаратного или программного прерывания.

Формула (FORMULA) – элемент, предназначенный для создания цепочки исполнения блоков с целью преобразования входных данных.

Алгоритм (ALGORITHM) – элемент, предназначен для агрегации (объединения) простых структурных единиц программы в одну более сложную систему. Под структурными единицами подразумевается FORMULA, несущая определённую функцию, STATE_FLOW (набор функций, из которых запускается только одна в зависимости от состояния). Внутри алгоритма можно создать сложную структуру с помощью блока условия IF. Также возможно создать множество вложений ALGORITHM, WHILE, STATE_FLOW.

Цикл (WHILE) – то же самое, что и ALGORITHM, но выход из системы созданной внутри этого блока может произойти только по заданному в параметрах блока условию. **Важно**, чтобы условие выхода из блока WHILE всегда срабатывало, иначе произойдёт зависание ядра.

Ветвление (IF) – внешний блок, предназначенный для разветвления последовательностей исполнения алгоритмов, имеет один вход и два выхода True и False. В зависимости от выполнения заданного в параметрах блока условия будут выполняться подключенные модули и системы к ветке True либо False.

State-машина (STATE_FLOW) – специальный блок, реализующий машину состояния – метод программирования и представления процессов определёнными состояниями, переход из одного состояния в другое происходит по заданному условию. Состояния внутри системы State-машина создаются блоками State и STATE_FLOW. На линии связи между блоками назначаются условия перехода.

Также можно сделать вложение STATE_FLOW, что позволяет создать машину состояния в машине состояния.

Внутри блока STATE_FLOW может быть добавлены блоки FORMULA. Схемы, собранные в блоках FORMULA будут выполняться, если выполняется STATE_FLOW, в котором они расположены.

Переменные VAR – внешний блок, добавляемый на главное поле набора. Каждый блок VAR выделяет ячейку памяти для хранения данных. Обращение (чтение, запись) к переменной осуществляется с помощью блоков IN и OUT.

Разрыв связи (далее телепорт TP_IN, TP_OUT) – инструмент для разрыва связи потоков данных в блоках FORMULA, STATE. Может применяться для уменьшения загромождения схемы, связи между частями программы и установкой связи между библиотеками.

BACKGROUND – ветка программы, которая выполняется в фоне работы процессора (все свободное от прерываний время процессора) в так называемом Idle Loop – цикле незанятости. К данной ветке программы необходимо подключать функции, не требующие строгой периодичности и не критичные к времени выполнения.

Hardware Interrupt – аппаратные прерывания в процессоре. Происходят со строгой периодичностью, предназначены для реализации режима реального времени. Назначаются на блок EVENT, прикрепленная к данному EVENT ветка программы будет выполняться на частоте прерывания.

Software Interrupt – программные прерывания, происходят при выполнении заданного условия.

Стартовый проект – программный проект, содержащий минимальный набор инициализации периферии, с включенным ядром MexBIOS kernel. Для процессоров фирмы TI представляет собой проект в среде программирования CCS3.1 или 4 (смотрите документацию к библиотеке).

Структура программы – конструкция из внутренних блоков, собранных по специальным правилам, которая определяет устройство программы, порядок и логику выполнения модулей программы.

Программа – совокупность всех добавленных в конкретный проект блоков. По программе создаётся матрица, которая зашивается в память чипа.

Чип – это процессор или микроконтроллер с предустановленным ядром - программным обеспечением MexBIOS Kernel и библиотекой блоков.

Обмен данными с чипом – режим, когда установлена связь с чипом и можно загрузить конфигурационную матрицу в память микроконтроллера и начать процесс обмена данными по каналу связи. При этом данные будут отображаться на графических элементах на поле набора и в окне **Переменные**. Возможно построение графиков с помощью осциллографа.

Таблица графических знаков

Кликнуть левой кнопкой мыши (ЛКМ)

Нажать и удерживать ЛКМ

Отпустить ЛКМ

Кликнуть два раза ЛКМ

Кликнуть правой кнопкой мыши (ПКМ)

Нажать и удерживать ПКМ

Отпустить ПКМ

Текст примечания. Существенное дополнение к изложенному материалу.

Знак Внимание. Существенная информация, раскрывающая особенности, которые необходимо учесть при использовании программы.

Невыполнение указанного примечания может привести к вероятности выхода из строя оборудования или нежелательным последствиям.

Интерфейс MexBIOS™ Development Studio

В данном разделе будет описан интерфейс программы. Разработчики стремились создать интуитивно понятный интерфейс, однако в области программирования микроконтроллеров существует своя специфика, поэтому рекомендуем ознакомиться со следующей главой данного руководства.

Если пользователь имеет опыт визуального программирования, то может начать знакомиться с программой с раздела **принципы создания программ**.

Примечание. Для работы с конкретным процессором необходимо установить библиотеку блоков для этого процессора (см. документацию к библиотеке).

Существует два языка интерфейса MexBIOS™ Development Studio: **русский** и **английский**. Для переключения между языками необходимо в главном меню выбрать пункт **Язык** и нужный язык. Далее приводится описание русского интерфейса программы.

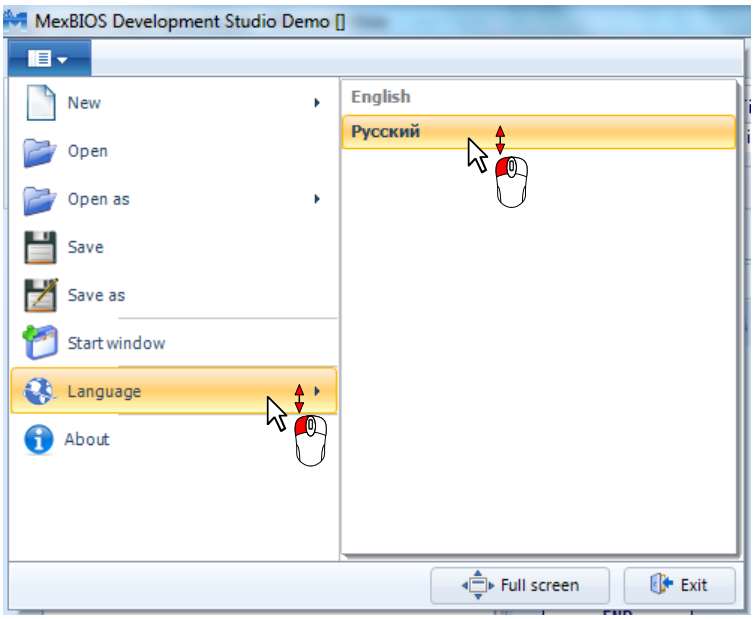


Рис. 1. Переключение языка интерфейса

Окно приветствия

При запуске MexBIOS™Development Studio по умолчанию отображается **Окно приветствия**. Внешний вид окна представлен на следующем рисунке:

Рис. 2. Вид стартового окна

В Окне приветствия доступны следующие действия:

- **Создать проект.** При нажатии на кнопку появится список из установленных библиотек, если нажать на один из пунктов – произойдёт создание нового файла проекта. После создания нового проекта, Стартовое окно закроется.
- **Открыть проект**. При нажатии на кнопку откроется диалог выбора файла проекта. После открытия выбранного проекта, Стартовое окно закроется.
- Список **Последние проекты**. Отображает список недавно открытых проектов. При первом запуске список пуст. После нажатия на проект из списка, произойдёт открытие проекта. Стартовое окно закроется.
- Кнопка **Руководство пользователя** откроет Руководство пользователя.
- Кнопка **Описание блоков** откроет документ с описанием блоков.
- Кнопка **Описание загрузки** откроет список установленных библиотек. При нажатии на соответствующий библиотеке пункт в списке, произойдёт открытие описание загрузки стартового проекта и начала работы с процессором этой библиотеки.
- Кнопка **Примеры** откроет примеры доступные для скачивания на сайте разработчика.
- Чтобы окно не отображалось при старте, необходимо убрать галочку **Отображать окно**

при запуске в левом нижнем углу Стартового окна.

- Чтобы закрыть Стартовое окно и перейти к Главному окну, нажмите кнопку **Закрыть** в правом нижнем углу.
- Чтобы заново вызвать **Окно приветствия** необходимо нажать по значку главного меню и выбрать значок стартового окна:

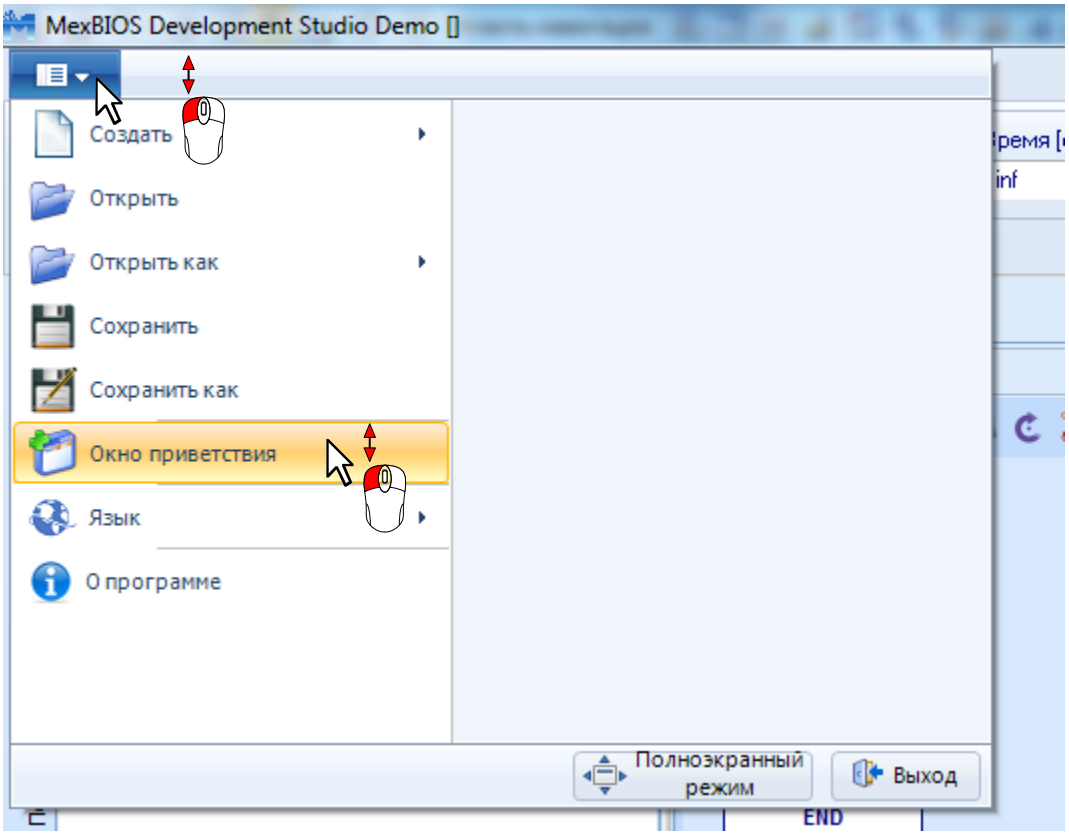


Рис. 3. Вызов Окна приветствия

Главное окно

Интерфейс программы состоит из закреплённых панелей, каждая из которых выполняет определённые функции. При запуске MexBIOS™Development Studio выглядит, как показано на рис. 4.

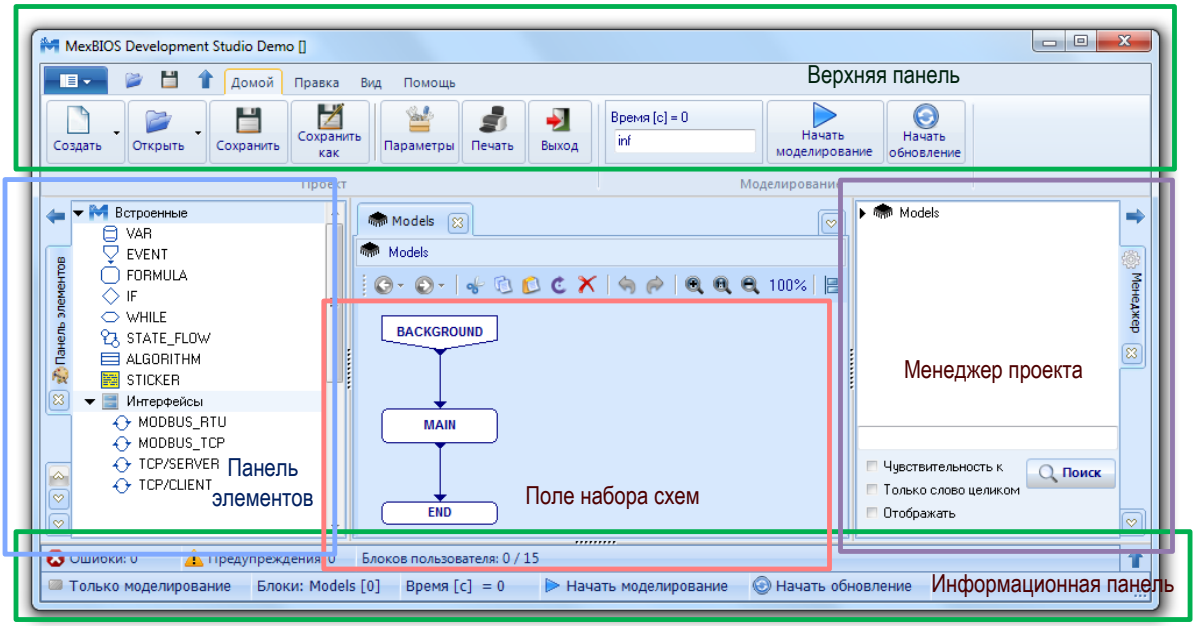


Рис. 4. Окно MexBIOS™Development Studio при первом запуске

Верхняя панель содержит кнопки для управления, редактирования, отображения проекта и др. Имеет структуру вкладок, каждая вкладка содержит набор кнопок предназначенных для выполнения определённых функций.

Панель элементов – отображает встроенные блоки и библиотеку внешних блоков моделей и выбранного процессора. Содержание панели элементов изменяется в зависимости от типа поля набора схемы.

Поле набора схем – представляет собой поле для набора схем из внутренних и внешних блоков. Внутренние блоки предназначены для формирования структуры программы, внешние блоки предназначены для создания программы.

Менеджер проекта – отображает древовидную структуру проекта. С помощью менеджера проекта можно осуществить поиск блока, переход между системами (формулами, алгоритмами и т.д.), создание закладок на схеме, расстановку очерёдности выполнения блоков, переход между используемыми осциллографами и комментариями.

Информационная панель – содержит сведения о режиме работы (моделирование или работа с чипом), число блоков, добавленных из библиотек на поле набора, пройденное время моделирования. Ошибки и предупреждения можно посмотреть в окне **Ошибки и Предупреждения**. Здесь же отображается информация о количестве созданных пользовательских блоков. Также, для удобства использования, на информационную панель помещены наиболее используемые кнопки.

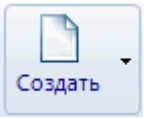
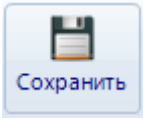
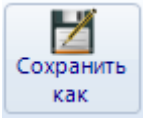
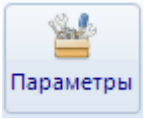
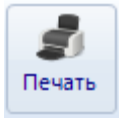
Описание вкладок главного меню

Домой

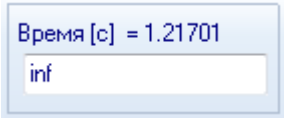
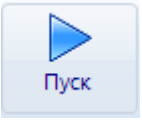
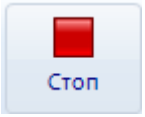
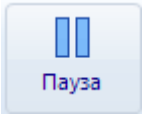
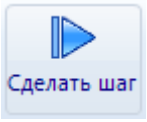
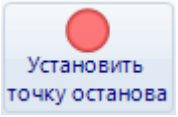
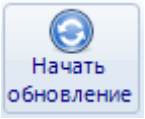
Вкладка основных действий над файлом проекта и управлением режимом моделирования.

Рис. 5. Содержание вкладки **Домой**

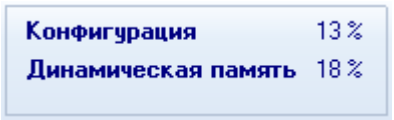
Группа Проект

	Создание нового файла проекта последней задействованной библиотеки. Для выбора другой библиотеки необходимо нажать на маленький треугольник ПКМ (появится список доступных библиотек) и выбрать нужную библиотеку, из установленных на компьютере
	Открытие существующего проекта
	Сохранить текущий проект
	Сохранить текущий проект под другим именем или заменить существующий проект
	Опции проекта. При нажатии на эту кнопку появится окно настроек проекта, которое включает в себя настройки: симуляции, соединения с устройством, дополнительные опции поля набора
	Печать текущей схемы на принтере
	Выход из программы

Группа Моделирование

	Отображение прошедшего времени моделирования и диалог задания конечного времени моделирования. Если установлен 0, то моделирование будет продолжаться, пока не будет произведена остановка нажатием на Стоп.
	Кнопки пуска моделирования. Произвести запуск моделирования можно с помощью нажатия на кнопку ЛКМ, либо воспользоваться горячей клавишей **F5**.
	Для остановки моделирования необходимо нажать на кнопку **Стоп**, либо воспользоваться горячей клавишей **Shift + F5**.
	В процессе моделирования можно остановить выполнение программы, для этого нужно нажать на кнопку **Пауза**. Для продолжения моделирования необходимо нажать кнопку **Пуск**, либо начать пошаговую отладку кнопкой **Сделать шаг**.
	Кнопка **Сделать шаг** предназначена для пошаговой отладки программы в режиме моделирования. При нажатии на эту кнопку программа выполнит один блок.
	Кнопка **Установить точку останова** позволяет установить точку останова на выделенном блоке. После установки точки останова можно нажать кнопку Пуск, программа выполнит один цикл и снова остановится на блоке с точкой останова, либо выполнится до следующей точки останова, назначенной на другой блок. **См. функция пошаговой отладки программы.**
	Продолжить обновление данных. В симуляции равнозначно **Начать моделирование**. Для остановки необходимо нажать кнопку **Остановить обновление**.

Группа Мониторы

	Монитор ресурсов, не отображается, если создан проект только для библиотеки **Models**. Показывает в процентном соотношении занятую память, которую будет занимать проект в памяти процессора. **Конфигурация (CFG MEM)** – память, нужная для записи конфигурационного файла в память процессора. **Динамическая память (HEAP MEM)** – память, занимаемая для параметров и адресов.

Устройство

Вкладка содержит кнопки для работы с устройством:

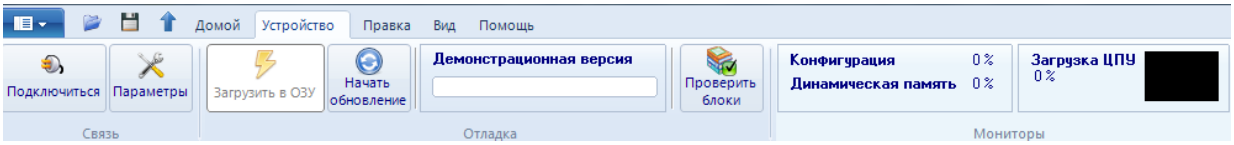
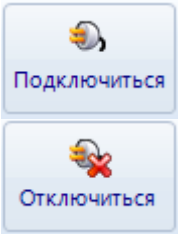
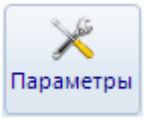


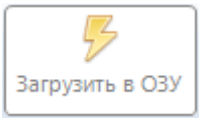
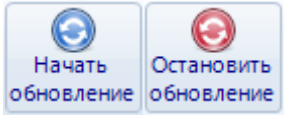
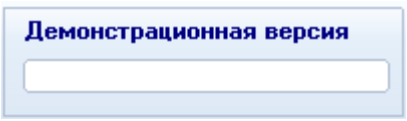
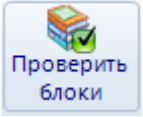
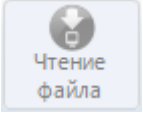
Рис. 6. Содержание вкладки **Устройство**

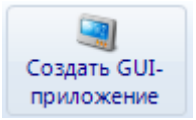
Вкладка **Устройство** появляется, только при активном проекте с библиотекой блоков процессора.

Связь

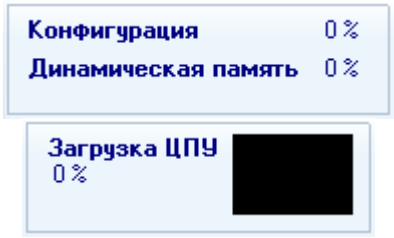
	Подключение / отключение связи с чипом. При удачном подключении значок изменится на **Отключиться**.
	Опции подключения. Настраивается способ подключения COM или USB. Для COM выбирается номер порта и общие настройки. Подробнее в настройках подключения смотрите в разделе «**Загрузка программы в память чипа**».

Отладка

	Загрузить конфигурационный файл в память контроллера. Кнопка активна при установленной связи с устройством.
	Начать / остановить обмен данными с чипом по выбранному порту в окне **Параметры** и каналу связи (если установлена связь с контроллером)
	Полоса прогресса загрузки матрицы в память контроллера. Также отображает лицензию, по которой работает программа.
	Проверка соответствия библиотеки блоков, загруженных в память процессора (необходимо чтобы с процессором была установлена связь), с используемой библиотекой. После проверки, при несовпадении уникального номера блока GUID, выдаст сообщение с именами блоков, которые не соответствуют текущей библиотеке. Блоки автоматически исключатся из палитры.
	Кнопка для вызова диалога считывания файла журнала из памяти контроллера.

	Кнопка создания GUI приложения (пользовательский интерфейс).

Мониторы

	**Конфигурация** – процент заполнения памяти отведённой под конфигурационный файл. **Динамическая память** – процент заполнения памяти отведённой под данные. **Загрузка ЦПУ** – монитор отображения загрузки процессора контроллера, работает только при установленной связи с МК, загруженным конфигурационным файлом и включенным обновлением данных.

Правка

Вкладка функций работы с блоками на панели набора и создания новых блоков;

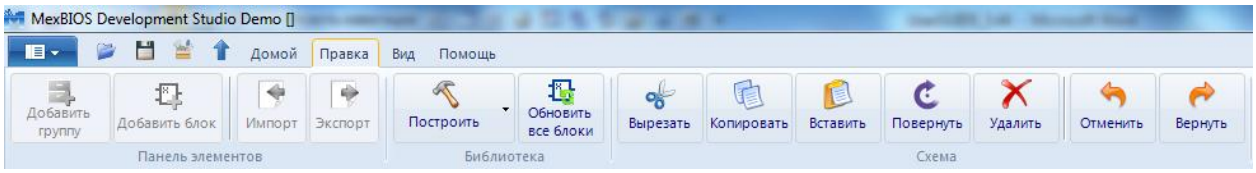
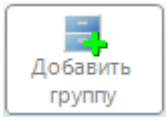
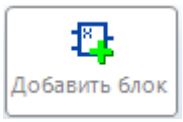
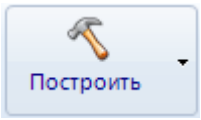
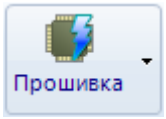
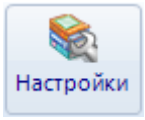
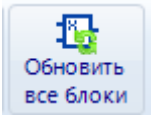


Рис. 7. Содержание вкладки **Правка**

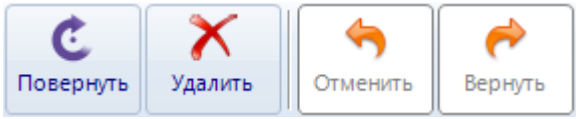
Панель элементов

	Добавить группу блоков в палитру. Кнопка активна только при выделенной библиотеке или группе в палитре. Такая палитра активна только на поле набора FORMULA, STATE.
	Добавить блок в группу. Кнопка активна только при выделенной библиотеке или группе в палитре блоков. Такая палитра активна только на поле набора FORMULA, STATE.
	Импортировать блок или группу блоков в палитру. Необходимо выбрать папку с файлами блока в открывшемся окне. Нажать ОК. Блоки появятся в группе, которая была выделена. Блоки скопируются в папку библиотеки: **C:\Users\%USERNAME%\AppData\Roaming\NPF Mechatronica-Pro\MexBIOS Development Studio\Extend\[БИБЛИОТЕКА]**
	Экспортировать блок или группу блоков из палитры. Необходимо выделить группу блоков или один блок. Нажать на кнопку Экспорт, появится диалог выбора пути, по которому будет сохранена группа блоков или блок. Нажать ОК. Блоки экспортируются в папку с именем библиотеки в подпапке с именем группы.

Библиотека

	Откомпилировать библиотеку. Содержит функцию Build All – откомпилировать все блоки и библиотеку. Для компиляции можно воспользоваться горячей клавишей **Ctrl+B**.
	Вызов окна для загрузки библиотеки блоков и стартового проекта в память контроллера. Для выбора утилиты загрузки нужно нажать ЛКМ на маленький треугольник. Подробней смотрите в разделе «**Загрузка библиотеки и стартового проекта в память контроллера**» **Примечание**: Для проекта библиотеки Models кнопка не отображается.
	Опции библиотеки. Путь к компилятору, распределение памяти. Подробней смотрите в разделе «**Настройка компиляции библиотек**»
	Кнопка **Обновить все блоки** запускает специальную функцию обновления всех блоков из палитры. Функция применяется, если в открытой схеме используются блоки, которые были изменены в текущей версии библиотеки.

Схема

	Кнопки «вырезать», «копировать», «вставить» элемент схемы.
	Повернуть элемент схемы по часовой стрелке, удалить выделенный элемент, отменить предыдущее действие, повторить отменённое действие

Вид

Вкладка основных инструментов MexBIOS™ Development Studio:

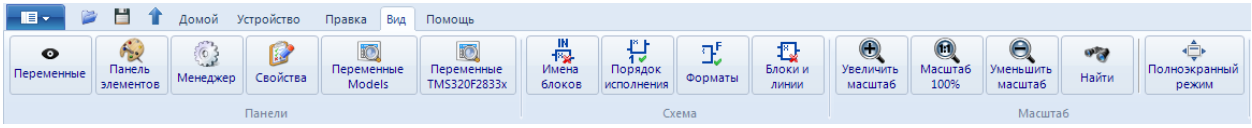
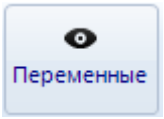
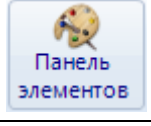
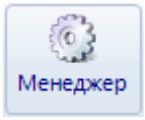
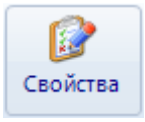
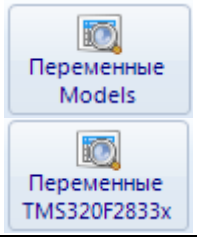
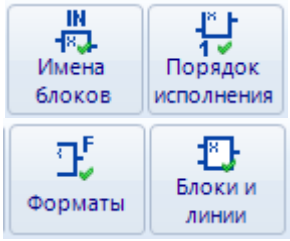
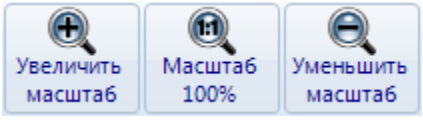
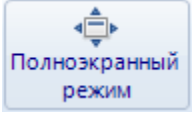
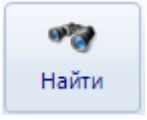


Рис. 8. Содержание вкладки Вид

Панели

	Отобразить окно Переменные на левой боковой панели
	Отобразить окно **Константы** на левой боковой панели
	Отобразить вкладку палитры блоков

	Отобразить менеджер проекта
	Отобразить инспектор блоков и свойства для выделенного блока
	Отобразить диалоговые окна Переменные Models и Переменные TMS. Подробней смотрите в пункте **Окно Таблица переменных**.
 Активна только на поле набора блоков	При работе со структурой программы кнопки не активны. Активация кнопок происходит при переходе в окно набора программ. **Имена блоков** – кнопка скрытия/отображения имён блоков; **Порядок исполнения** – кнопка отображения/скрытия индексов вызова блоков (порядок вызова); **Форматы** – кнопка отображения форматов входов и выходов блоков; **Блоки и линии** – кнопка скрытия всех блоков и линий, отображение только лишь органы управления.
	Инструменты масштабирования схемы. Увеличить масштаб – увеличить текущую схему; Уменьшить масштаб – уменьшить текущую схему; Масштаб 100% - вернуть первоначальный масштаб.
	Развернуть рабочую область на полный экран.
	Поместить выделенный элемент по центру экрана.

Помощь

Справка

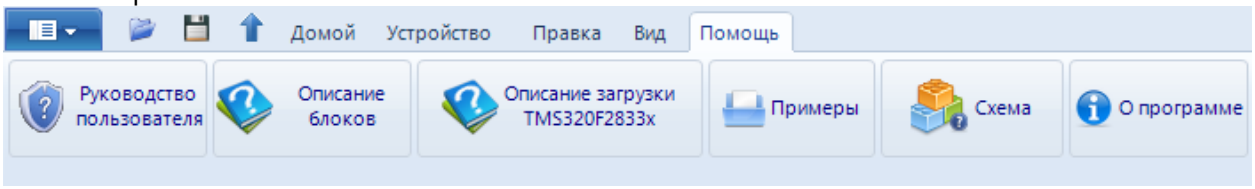
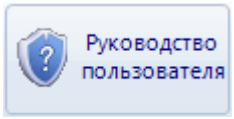
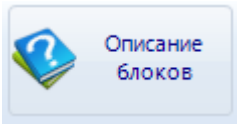
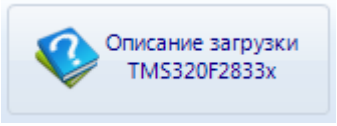
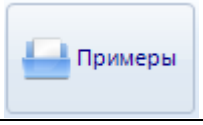
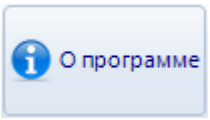


Рис. 9. Содержание вкладки Помощь

	Справка по применению среды MexBIOS™ Development Studio;
	Вызов справки по библиотеке блоков.
	Вызов справки по загрузки стартового проекта и библиотеки блоков в память процессора выбранной библиотеки.
	Отрывает примеры на сайте.
	Вызов справки по текущей схеме. При нажатии этой кнопки вызывается pdf файл с тем же названием, что и у открытой схемы из каталога, откуда была открыта схема. Если файла не существует, выдаётся сообщение «Нет связанного файла документации!»
	Вызывает окно **О программе**, которое отображается при загрузке MexBIOS™ Development Studio

Для начала работы с конкретной библиотекой блоков необходимо нажать кнопку главного меню на верхней панели, в пункте **Создать** выбрать одну из установленных библиотек блоков для процессоров, например **TMS320F2833x**.

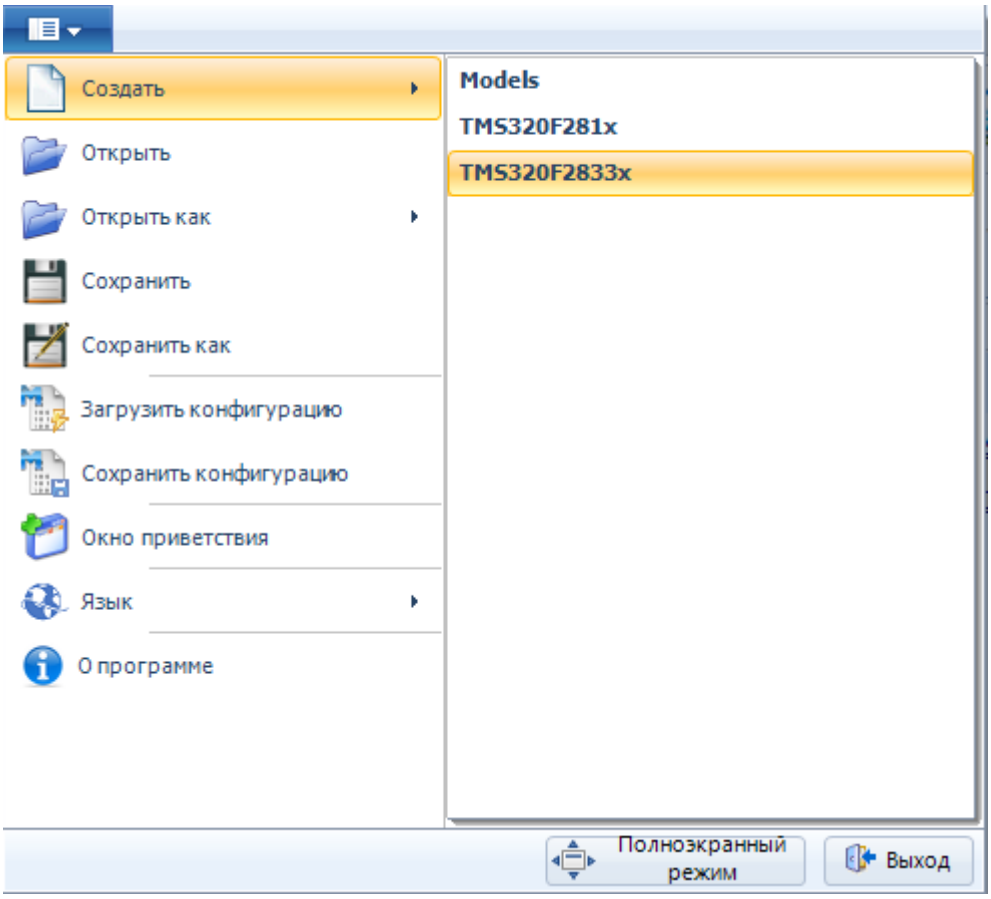


Рис. 10. Главное меню, выбор библиотеки блоков для нужного процессора

Если были произведены какие-либо изменения в открытой схеме - появится диалоговое окно запроса сохранения изменений, если схему не нужно сохранять выбрать «Нет», при необходимости сохранить схему выбрать «Да».

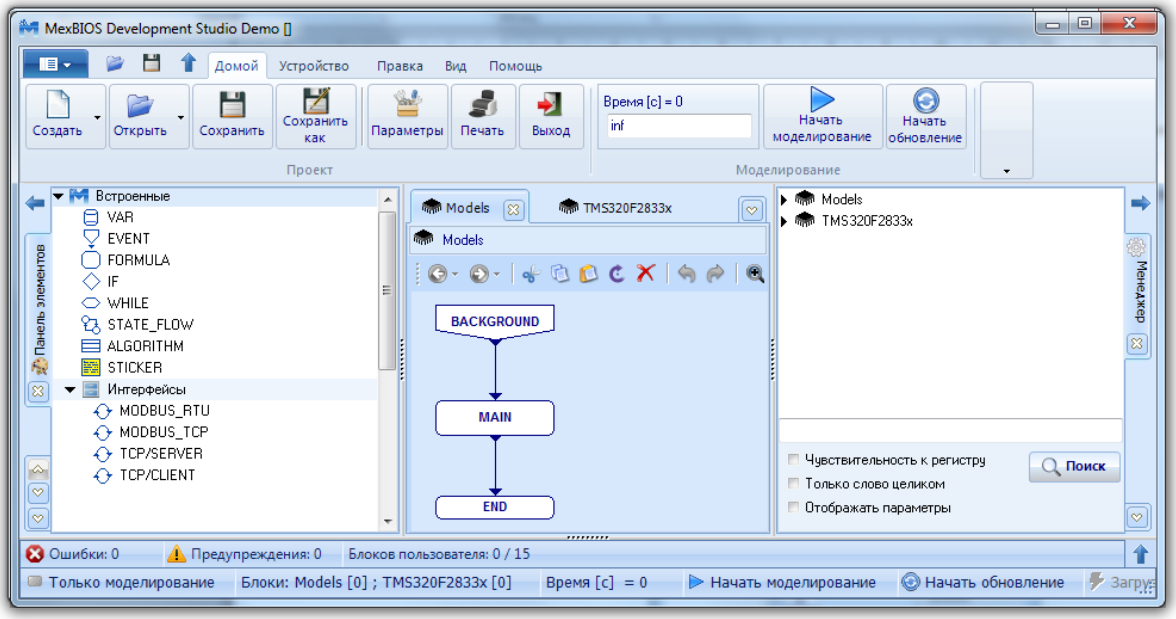


Рис. 11. Пример окна нового проекта для TMS320F2833x

Работа с окнами

Главное окно, с раскрытыми левой и верхней панелями и скрытой правой панелью, выглядит, как показано на рис. 12. По умолчанию создаётся файл библиотеки **Models**.

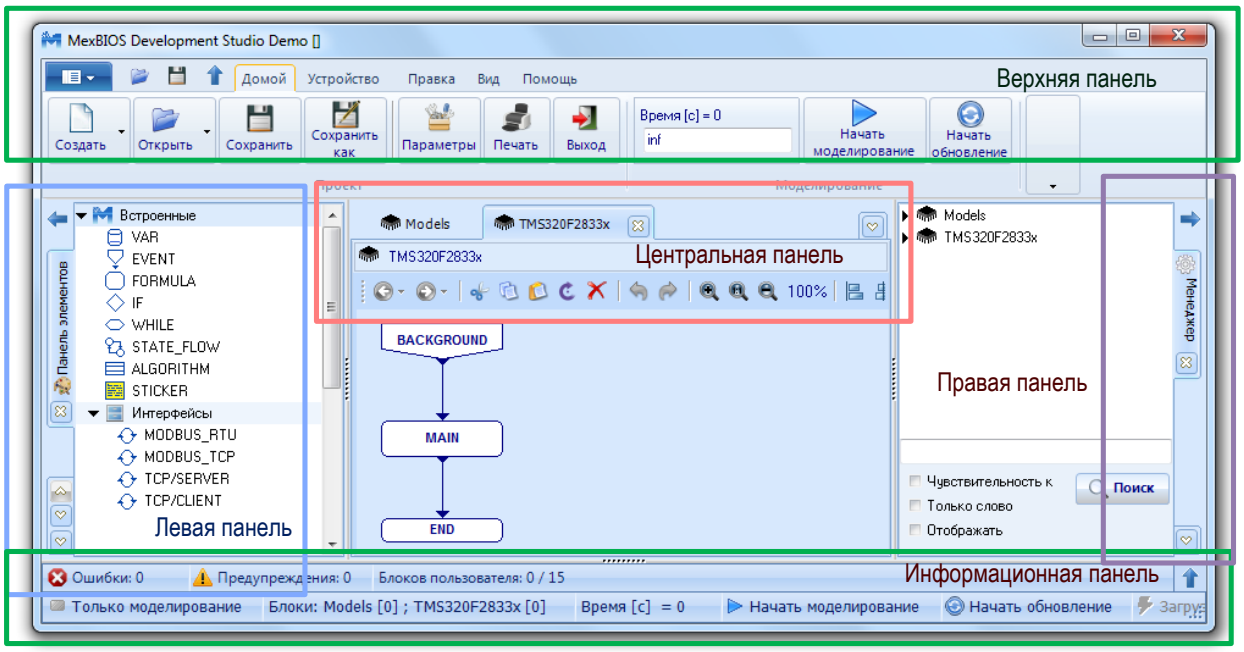


Рис. 12. Окно MexBIOS™Development Studio при созданном новом проекте TMS320F281x

Верхнюю, левую и правую панели при желании можно скрыть. Для этого в ряду с заголовками страниц на панели располагаются кнопки с пиктограммой в виде стрелки, показывающей направление скрытия раскрытия панели  .

Также панели можно раскрывать нажатием на заголовки закладок встроенных в панель.

Для перехода между вкладками необходимо кликнуть ПКМ по названию вкладки. По умолчанию в панелях имеются следующие закладки:

Левая панель содержит две вкладки:

- **Панель элементов** – палитра блоков. В зависимости от открытия того или иного поля набора, будут отображаться блоки доступные для вынесения на поле набора из палитры.
- **Переменные** – инструмент отображения и изменения данных при моделировании и работы с процессором;

На центральной панели размещены следующие элементы:

- **Строка пути**. Срока отображает путь, открытой схемы набора. Можно нажать на любую позицию из пути и перейти на другое поле набора.
- **Панель редактирования схемы, которая содержит:** кнопки вперёд/назад по схеме (также содержат историю переходов); кнопки вырезать, скопировать, вставить, повернуть, удалить выделенный блок; кнопки отмены/повтора текущего действия; масштаб схемы; вызов инспектора.
- **Models** – главное поле набора для библиотеки Models;
- **TMS320F2833x** – главное поле набора для файла библиотеки процессора. Название вкладки зависит от названия открытой библиотеки блоков;
- **Вкладки**, открываемые из поля набора схемы.

Правая панель содержит две вкладки:

- **Менеджер** – инструмент для перехода между формулами и алгоритмами, добавленными в программу, а также расстановки порядка очерёдности выполнения блоков в проекте;
- **Свойства** – инструмент отображения параметров выбранного блока, отображается при двойном нажатии ЛКМ по блоку.

У левой (Панель элементов, Переменные), центральной (Models, TMS320F281x) и правой панели (Менеджер, Свойства), у всех кроме верхней (далее речь только о панелях, кроме верхней), можно менять состав и порядок вкладок. Любую из вкладок можно отсоединить от родительской панели и превратить в самостоятельное окно, а затем снова встроить в любую из панелей. Для того чтобы превратить вкладку в самостоятельное окно достаточно двойного нажатия левой кнопкой мыши на заголовке соответствующей вкладки.

При захвате ЛКМ заголовка отсоединённой вкладки посередине экрана будет появляться значок:

Рис. 13. Кнопки закрепления окна

Для закрепления вкладки на левой, центральной или правой панели необходимо захватить заголовок окна ЛКМ, затем перетащить курсор мыши на соответствующий значок рис. 13, например, для закрепления на среднюю панель, и отпустить ПКМ. Вкладка закрепится на выбранной панели:

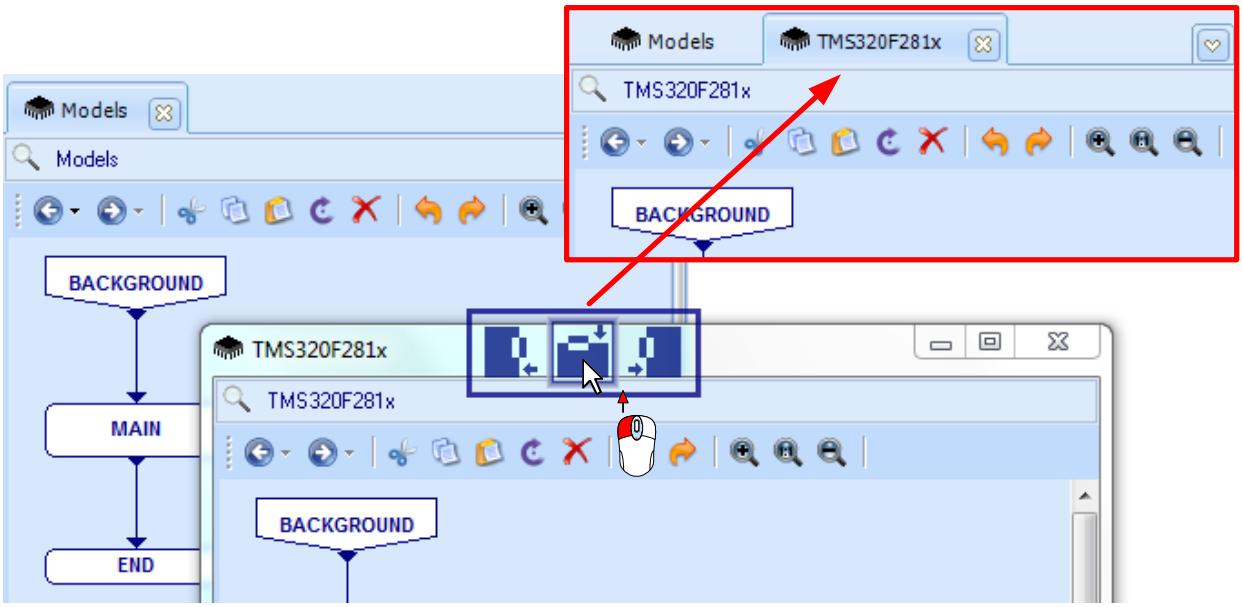


Рис. 14. Закрепление вкладки

Работа с элементами схемы

В данном разделе будет рассмотрена особенность создания структуры программ, методы работы с блоками. Блоки из палитры можно перетащить мышью на схему набора, тем самым создав экземпляр блока. У внутренних и внешних блоков есть своя область применения. Для разработки программного обеспечения существует ряд правил, которые можно посмотреть во введении, см. раздел «**Ключевые правила для разработки программного обеспечения в ИС**».

Панель элементов

Внешний вид

Палитра состоит из разделов в виде списков и блоков сгруппированных по разделам палитры. Список **Embedded** включает в себя встроенные блоки, которые не могут быть отредактированы. Палитра изменяется, в зависимости от области применения блоков. Всего может быть пять различных состояния палитры.

1. Открыто главное поле набора структуры программы.
2. Открыто поле набора STATE_FLOW.
3. Открыто поле набора ALGORITHM.
4. Открыто поле набора схемы программы (поле набора внутри блоков FORMULA и STATE).
5. Открыто поле набора SUBSYSTEM.

Из палитры, перетаскиванием мыши, производится вынос блоков на открытое поле набора.

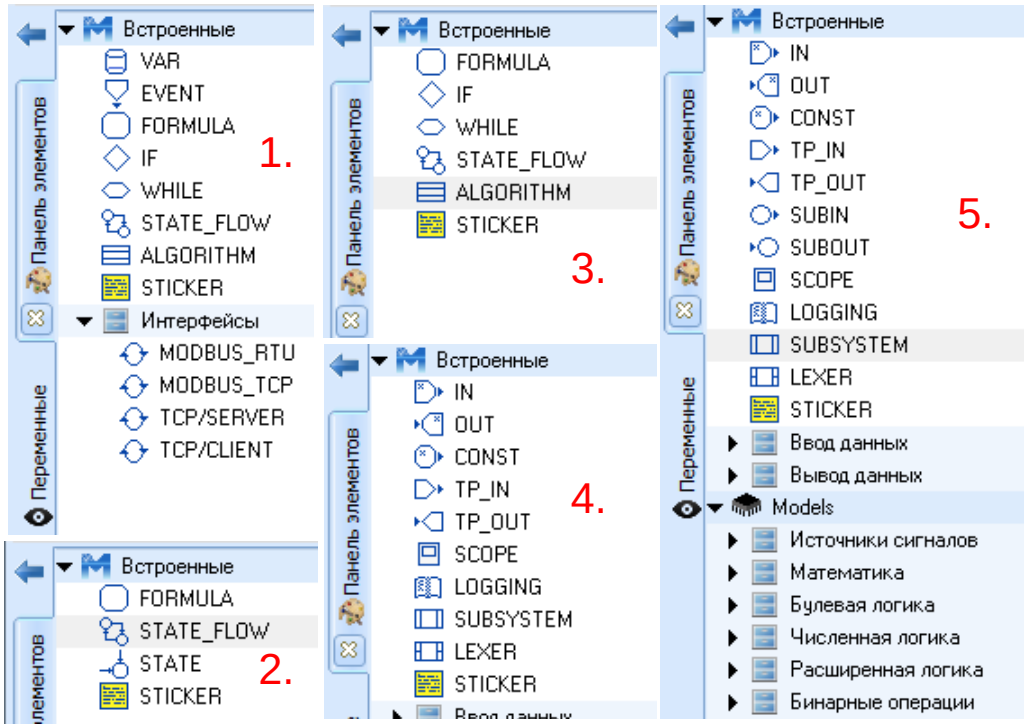


Рис. 15. Палитра

Редактирование внешних блоков описано в разделе «**Работа с библиотеками блоков**».

При несоответствии библиотеки блоков на компьютере с библиотекой блоков зашитой в

контроллере (нет блока или не совпал GUID-номер), происходит запрет для использования блоков в схемах. Подробней о проверке соответствия библиотек см. **Функция верификации библиотеки**.

Рис. 16. Запрещенные блоки

Проверка соответствия блоков и разрешение использования запрещенных блоков описаны в разделе «**Функция верификации библиотеки**».

Описание встроенных блоков Embedded

EVENT	Событие
Назначение	Запускает, подключенные к событию, системы (ALGORITHM, FORMULA, WHILE, STATE_FLOW), в зависимости от настроек: на программном прерывании либо на аппаратном прерывании.
Группа	**Встроенные**
Изображение блока	
Настройки	Параметр **Source** (источник прерывания) может быть **Аппаратным** и **Программным**. Для **Аппаратного прерывания** следующие настройки: **Вектор** – выбор вектора прерывания, настроенного в стартовом проекте. **Период** – настройка периода выполнения, необходимая для симуляции программы. Должна быть равна периоду аппаратного прерывания. **Режим моделирования** – режим моделирования прерывания на компьютере. Выключено – прерывание не выполняется, Однократное – выполниться один раз при пуске, Непрерывное – будет выполняться непрерывно. Для **Программного прерывания**: **Условие** – условие запуска **Значение 1** – величина 1, может быть как VAR, так и константой. **Значение 2** – величина 2, может быть как VAR, так и константой. **Формат** – задание формата данных, которые используются для сравнения в режиме программного прерывания. По умолчанию формат Integer. Внимание: Срабатывание программного прерывания может работать некорректно, если указан формат данных не совпадающий с форматом переменных указанных в Значение 1 и Значение 2.

Описание	Блок выносится из палитры на поле набора. Имеет один выход. К выходу подключаются вход другого внутреннего блока. **Можно подключить только один блок.** Программные прерывания выполняются в фоне процессора и не имеют строгой периодичности. В качестве события Программные прерывания можно использовать выполнение условия сравнения 2х переменных. Моделирование на компьютере производиться с заданным периодом.  **Внимание: Используемые в проекте аппаратные прерывания, должны быть настроены и апущены в стартовом проекте соответствующим образом.** **Примечание:** Настройка **Режим моделирования** имитирует выполнение прерывания в процессе моделирования. Для работы в контроллере, выбранное аппаратное прерывание должно быть разрешено в стартовом проекте. Если прерывание не запущенно в стартовом проекте, то ветка программы, присоединённая к EVENT, выполняться не будет.
Область действия	Блок может быть добавлен только на главное поле набора.

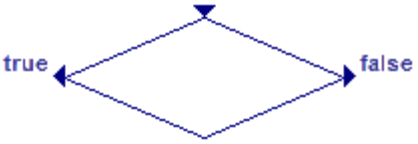
ALGORITHM	Алгоритм

Назначение	Система, внутри которой может быть построена схема из других встроенных блоков (кроме EVENT). Блок предназначен для создания обособленной части программы.
Группа	**Встроенные**
Изображение блока	
Описание	Блок выносится из палитры на поле набора. Имеет один вход и один выход. К входу необходимо напрямую подключить блок EVENT, либо подключиться через цепочку других блоков, подключенных к блоку EVENT. К выходу можно последовательно присоединить другие внутренние блоки.
Область действия	Блок может быть добавлен на поле набора как для Models, так и для открытой библиотеки чипа. Также можно создать вложенность одного ALGORITHM в другой. Внутри блока могут содержаться блоки IF, FORMULA, ALGORITHM, STATE_FLOW, STICKER.

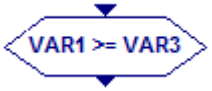
FORMULA	Формула

Назначение	Внутри блока создаётся функциональная часть реализуемой программы управления из внешних блоков, расположенных в палитре.
Группа	**Встроенные**
Изображение блока	
Описание	Блок выносится из палитры на поле набора. Имеет один вход и один выход. К входу необходимо напрямую подключить блок EVENT, либо подключиться через цепочку других блоков, подключенных к блоку EVENT. К выходу можно последовательно присоединить другие внутренние блоки. Из FORMULA можно передать сигнал с помощью TP_IN, TP_OUT в другую FORMULA, либо STATE, либо FORMULA и/или STATE библиотеки MODELS.
Область действия	Блок может быть добавлен на поле набора как для Models, так и для библиотеки процессора. Внутри блока могут содержаться блоки TP_IN, TP_OUT, STICKER и внешние блоки к открытой библиотеке.

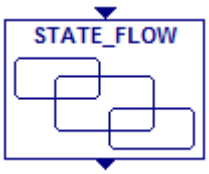
IF	Условие «Если»

Назначение	Разделение алгоритма программы на два возможных случая, в зависимости от выполнения назначенного условия
Группа	**Embedded**
Изображение блока	
Настройки	Параметр Condition – логическое условие сравнения либо оператор применяемый к **Значение 1** и **Значение 2**. Доступны логические операции: равно (==), не равно (!=), больше(>), больше либо равно (>=), меньше (<), меньше либо равно (<=), И (AND), И-НЕ(AND-NOT), ИЛИ (OR), ИЛИ-НЕ (OR-NOT), исключающая ИЛИ (XOR), исключающая ИЛИ-НЕ (XOR-NOT). **Значение 1** и **Значение 2**– список добавленных на схему переменных. Сравнение переменной можно производить с константой, для этого необходимо выбрать пункт **Константа…**. **Формат** – выбирается формат данных, в котором выполняются операции.
Описание	Для работы с блоком IF необходимо указать условие выполнения ветки true или false. В зависимости от выполнения условия срабатывают системы (FORMULA, ALGORITHM, STATE_FLOW) подключенные последовательно к выходам блока. Допустимо оставлять одну пустую (не содержащую Формулу или Алгоритм) ветвь true или false.  Для корректной работы блока необходимо замкнуть ветки **true** и **false** в узел. К получившемуся узлу можно подключать другие внутренние блоки. После назначения условия внешний вид блока отображает назначенное условие: 
Область действия	Блок может быть добавлен на поле набора как для Models, так и для библиотеки процессора.

WHILE	Условие «Пока»

Назначение	Выполняет находящуюся внутри блока схему, пока выполняется заданное условие
Группа	**Встроенные**
Изображение блока	
Настройки	Параметр Condition – логическое условие сравнение либо оператор применяемый к **Value 1** и **Value 2**. Доступны логические операции: равно (==), не равно (!=), больше (>), больше либо равно (>=), меньше (<), меньше либо равно (<=), И (AND), И-НЕ(AND-NOT), ИЛИ (OR), ИЛИ-НЕ (OR-NOT), исключающая ИЛИ (XOR), исключающая ИЛИ-НЕ (XOR-NOT). **Value 1** и **Value 2** – список добавленных на схему переменных. Сравнение переменной можно производить с константой.
Описание	Для работы с блоком WHILE необходимо указать условие пока будет выполняться собранная внутри схема. После назначения условия внешний вид блока отображает назначенное условие:   **Внимание**: Необходим гарантированный выход из цикла WHILE за конечное число расчётов программы, иначе произойдёт потеря режима реаль□ого времени и зависание процессора.
Область действия	Блок может быть добавлен на поле набора как для Models, так и для библиотеки процессора.

STATE_FLOW	State-машина

Назначение	Блок реализует оболочку для создания State-машины
Группа	**Встроенные**
Изображение блока	
Описание	Блок STATE_FLOW может содержать вложение блоков STATE_FLOW, что позволяет создать state-машину внутри state-машины. Количество вложений ограничено количеством памяти, выделенной в контроллере. Внутри каждого блока STATE_FLOW можно создать блок FORMULA, причем блок FORMULA не будет иметь входа и выхода, что означает, что схемы собранные внутри блока будут выполняться всегда, если выполняется состояние, в котором расположена FORMULA. Внутри блока с помощью блоков STATE и STATE_FLOW создаётся машина состояния – действие программы описываются конкретными состояниями системы, и задаются условия перехода между этими состояниями (назначается на соединительные линии). Для блока STATE_FLOW существует настройка Start state, которой устанавливается начальное состояние, с которого начнётся выполняться State-машина. **Примечание:** По умолчанию стартовым состоянием назначается первое состояние STATE, помещённое на поле набора. Если схема собрана только блоками STATE_FLOW, то необходимо вручную задать стартовое состояние для родительского STATE_FLOW.
Область действия	Блок может быть добавлен на главное поле набора, ALGORITHM. Внутри блока могут находиться блок STATE, STATE_FLOW, FORMULA и STICKER

STATE	Состояние

Назначение	Блок состояния, внутри могут находиться внешние блоки из палитры. Блок может подключаться к другому блоку состояния и на линию связи назначается условие перехода из текущего состояния в другое состояние. Начальное состояние по умолчанию назначается первое, вынесенное на поле набора, состояние.
Группа	**Встроенные**
Изображение блока	
Описание	Внутри блока могут быть добавлены внешние блоки из палитры и блоки TP_IN, TP_OUT, STICKER.
Область действия	Блок может быть добавлен только внутрь блока STATE_FLOW

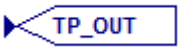
FORMULA	Формула для STATE_FLOW

Назначение	Внутри блока создаётся функциональная часть реализуемой программы управления из внешних блоков, расположенных в палитре. Формула выполняется в том случае, если активно состояние, в котором размещена формула.
Группа	**Встроенные**
Изображение блока	
Описание	Блок выносится из палитры на поле набора STATE_FLOW. Из FORMULA можно передать/использовать сигнал с помощью TP_IN, TP_OUT.
Область действия	Блок может быть добавлен на поле STATE_FLOW. Внутри блока могут содержаться блоки TP_IN, TP_OUT, STICKER и внешние блоки к библиотеке.

STICKER	Комментарий

Назначение	Блок создания комментариев на схеме. Хранит в себе текст комментария.
Группа	**Встроенные**
Изображение блока	
Описание	В параметр Strings записывается текст комментария
Область действия	Может находиться в любой из систем

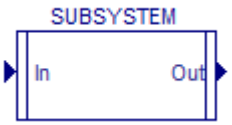
TP_OUT	Передатчик сигнала (Teleporter From)

Назначение	Блок позволяет передать сигнал из схемы, в которой он расположен, в любую другую схему в которой расположен блок TP_IN. Может связывать библиотеки Models и библиотеку блоков процессора (необходимо при моделировании).
Группа	**Встроенные**
Изображение блока	
Описание	Блок предназначен для передачи сигнала, присоединённого к входу блока, в блок TP_IN. Сигнал будет передаваться в любой (может несколько) привязанный к блоку TP_OUT блок TP_IN. Блоки TP_OUT позволяют организовать передачу данных между библиотеками и между использованными в программе встроенными блоками. Блок имеет параметр Tag, который отвечает за привязку к блоку блоков TP_OUT.  **Примечание:** В проекте не может быть двух блоков с одинаковыми Тег.
Область действия	Блок может находиться только в системах FORMULA и STATE.

TP_IN	Приёмник сигнала (Teleporter To)

Назначение	Блок приёмник сигнала из блока TP_OUT.
Группа	**Встроенные**
Изображение блока	
Описание	Блок предназначен для приёма сигнала из блока TP_OUT. Для связи блока TP_OUT с TP_IN, в настройках блока необходимо выбрать свойство Tag блока TP_OUT (от которого передаём сигнал) из выпадающего списка.
Область действия	Блок может находиться только в системах FORMULA и STATE.

SYBSYSTEM	Подсистема

Назначение	Создание подсистемы для внутренних блоков на поле набора FORMULA или STATE
Группа	**Встроенные**
Изображение блока	
Описание	При двойном нажатии ЛКМ на блоке откроется поле набора подсистемы, где можно собирать схему, либо вырезать часть большой схемы и вставить внутрь подсистемы. В палитре появится два блока SUBIN и SUBOUT – блоки входа и выхода для подсистемы. Дополнительные входа/выхода можно добавить из **Панели элементов** или скопировать существующие.
Область действия	Блок может находиться только в системах FORMULA и STATE.

Пункт PROTOCOLS содержит блоки поддерживаемых протоколов передачи данных. Подробней смотрите в разделе «**Блоки организации коммуникаций**».

Блок LEXER

Блок **Lexer** представляет собой специальный блок, в котором можно писать программу на языке подобному языку стандарта МЭК 61131-3 структурированный текст (ST). Не описанные далее структуры языка **ST** в блоке **Lexer** не поддерживаются. Также имеются особенности синтаксиса ветвления **IF** и цикла **WHILE**.

Рис. 17. Внешний вид блока при добавлении на поле набора

Для библиотек, поддерживающих вычисления в формате Float (например: TMS320F2833x, LM4F23 и др.) формат данных по умолчанию в блоке **Lexer** является Float. Для библиотек, которые производят вычисления в формате с фиксированной запятой, формат для блока **Lexer** по умолчанию **Integer**.

Данный блок поддерживает возможность обмениваться данными (считывать/записывать) с переменными, объявленными в проекте.

Внимание: Особенностью блока является то, что он должен содержать всегда правильный код. Редактор блока не закроется, пока текст программы не будет содержать ошибок.

При двойном нажатии на блок появится окно, представленное на следующем рисунке:

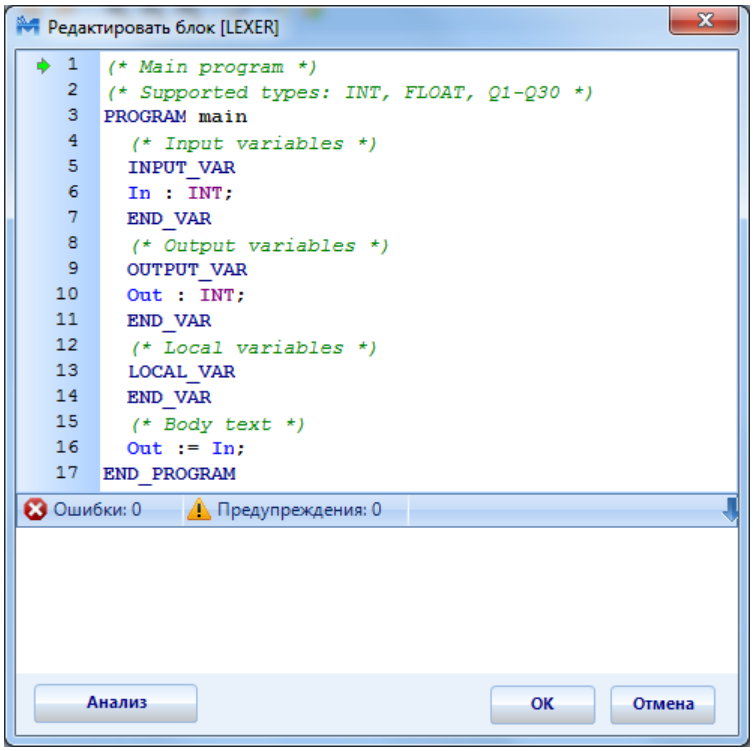


Рис. 18. Внешний вид раскрытого блока Lexer

Структура любой LEXER-программы имеет следующие зарезервированные слова:

PROGRAM – объявление начала программы. После этого ключевого слова должно

следовать имя программы на латинице. Имя не должно начинаться с цифры.

INPUT_VAR – после этого ключевого слова объявляются входа блока. В зависимости от числа объявленных входов в этой секции, будут появляться входа у блока на поле набора. Секция должна обязательно содержать ключевое слово **END_VAR**, означающее, в данном случае, завершение объявление входов.

OUTPUT_VAR – ключевое слово начала объявления выходов блока. В зависимости от числа объявленных выходов в этой секции, будут появляться выхода у блока на поле набора. Раздел должен обязательно содержать команду **END_VAR**, означающую, в данном случае, завершение объявление выходов.

LOCAL_VAR – раздел объявление локальных переменных, действующих внутри блока.

Далее следует основная программа, которая должна обязательно заканчиваться командой **END_PROGRAM**.

Комментарии заключаются в символы `(* *).`

Объявление входов/выходов

Объявление входов/выходов имеет следующий синтаксис:

Входа

```
(* Input variables *)
INPUT_VAR
In  : INT;
In1 : Q24;
In2 : FLOAT;
END_VAR
```

Выхода

```
(* Output variables *)
OUTPUT_VAR
Out  : INT;
Out1 : Q24;
Out2 : FLOAT;
END_VAR
```

Для каждого входа/выхода необходимо задать формат, в котором будут производиться вычисления. Соответственно на вход блока необходимо подавать сигнал в указанном формате. На выходе блока будет сигнал указанного формата.

Возможно использовать следующие форматы **INT** – целочисленный формат, **FLOAT** – формат с плавающей запятой, **Q1 – Q30** – формат фиксированной запятой.

Объявление локальных переменных

Объявление локальных переменных имеет подобный синтаксис, как и для входов/выходов, только возможно задать начальное значение:

```
(* Local variables *)
LOCAL_VAR
VAR  : INT   := 10;
VAR1 : Q24   := 12;
VAR2 : FLOAT := 0;
END_VAR
```

Если переменная объявления в формате VAR1 : Q24 := 12; Число в VAR1 запишется в формате Q24, дополнительно переводить не нужно. Также с другими форматами.

Использование внешних переменных

Если на главном поле набора объявлены переменные (вынесены из **Панели элементов** блоки **VAR**), то в тексте программы **Lexer** возможно непосредственное обращение к этим переменным.

Математические операции

Поддерживаются основные математические операции:

a * b - умножение;
a / b - деление;
a + b - арифметический плюс;
a -- b - арифметический минус;
-a – унарный минус;
+a – унарный плюс;

Внимание: Математические операции выполняются в формате, который задан по умолчанию для текущей библиотеки. Для библиотек поддерживающих вычисления с плавающей запятой это **Float**, для библиотек производящих вычисления в формате с фиксированной запятой это **Integer**.
Для математических вычислений в формате, отличном от формата по умолчанию необходимо использовать блоки как функции из группы **Math** (см. далее).

Использование внешних блоков как функций:

Можно использовать внешние блоки как функции. Например, для умножения в формате IQN необходимо написать следующее:

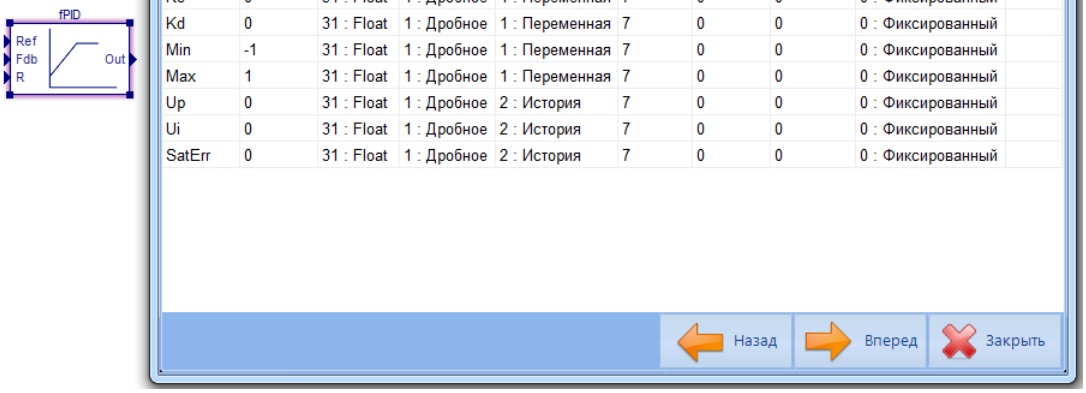
```
Out := MPY(VAR,Q24(2),24);
```

Где MPY – блок в группе **Math**. Необходимо указывать следующие аргументы для блока – входа и дополнительный параметр – формат.

Если у блока больше входов и параметров, то аргументами функции будут сначала входа сверху вниз, затем параметры, которые отображаются на вкладке **Параметры** в окне **Редактора блока**.

Свойства.

Рассмотрим PID регулятор:



Название	Значение	Формат	Редактор	Тип	Точность	Минимум	Максимум	Тип формата	Строки
Kp	1	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Ki	0	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Kc	0	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Kd	0	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Min	-1	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Max	1	31 : Float	1 : Дробное	1 : Переменная	7	0	0	0 : Фиксированный	
Up	0	31 : Float	1 : Дробное	2 : История	7	0	0	0 : Фиксированный	
Ui	0	31 : Float	1 : Дробное	2 : История	7	0	0	0 : Фиксированный	
SatErr	0	31 : Float	1 : Дробное	2 : История	7	0	0	0 : Фиксированный	

Рис. 19. Блок **PID** и его свойства.

Для использования функции **PID** регулятора в блоке **Lexer** необходимо написать следующее выражение:

```
Out := PID(Ref,Fdb,Res,1,0,0,0,-1,1, 0, 0, 0);
```

где Ref, Fdb, Res – входа блока **Lexer**, объявленные в поле ключевого слова **INPUT_VAR**.

Полученный блок **Lexer** будет иметь вид:

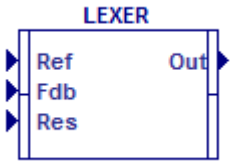


Рис. 20. Внешний вид блока Lexer

Приведение формата для констант

Для констант возможно применение преобразование формата:

```
FLOAT(3.14)
INT(1000)
Q1(3.5)
...
Q30(1.234)
```

Преобразование формата для констант можно использовать как аргументы функций, в сравнениях.

Внимание: Недопустимо использовать преобразование форматов для входов/выходов и для переменных.

Ветвление IF и цикл WHILE

Возможно использование структуры IF и цикла WHILE.

Пример ветвления IF:

Ветвление используется для проверки условия и в зависимости от этого условия выполнить какие-либо действия.

```
IF VAR > 0 THEN
VAR  := VAR + 1;
END_IF
```

Также можно использовать дополнительные условия:

```
IF VAR > 1000 THEN
VAR  := VAR + 1;
ELSE_IF VAR < 0 THEN
VAR := VAR + 2;
ELSE   VAR := VAR + 3;
END_IF
```

Можно использовать множество условий ELSE_IF.

Пример цикла WHILE:

Цикл выполняется, пока верно заданное условие. Необходимо, чтобы выход из цикла гарантированно выполнялся.

```
WHILE VAR < 10 THEN
VAR  := VAR + 1;
END_WHILE
```

Свойства блока

Инспектор предназначен для отображения и редактирования параметров выделенного блока. Отображает все доступные для редактирования сведения о блоке. Содержит два раздела. Первый раздел **Дизайн** предназначен для редактирования графических свойств блока: имени, положения на поле набора, размеров, цвета, ориентации. Второй раздел **Исполнение** отображает доступные для редактирования параметры функции блока библиотеки.

Для вызова инспектора необходимо два раза нажать по блоку ЛКМ либо a) с выделенным блоком нажать клавишу **F11**; b) на вкладке Вид нажать кнопку **Свойства**; c) нажать клавишу инспектора на центральной панели; d) нажать ПКМ по блоку и в контекстном меню выбрать первый пункт **Свойства**.

Внешний вид

Окно инспектора предназначено для отображения и редактирования параметров и свойств блока. Для отображения свойств блока в инспекторе и доступа к редактированию необходимо блок выделить и вызвать инспектор (горячая клавиша **F11**).

Дизайн	
Имя	PID
Смещение слева	110
Смещение сверху	230
Ширина	100
Высота	60
Цвет закрашивания	0xFFFFFF
Цвет линии	0x800000
Вид заполнения	0 : Сплошная заливка
Вид линии	0 : Сплошная линия
Толщина линии	1
Отображать имя	1 : Да
Направление	2 : Справа
Исполнение	
Format	24 : Q24
Kp	1
Ki	0
Kc	0
Kd	0
Min	-1
Max	1

Рис. 21. Окно Свойства

Группа свойств Дизайн

К этой группе относятся настройки внешнего вида блоков.

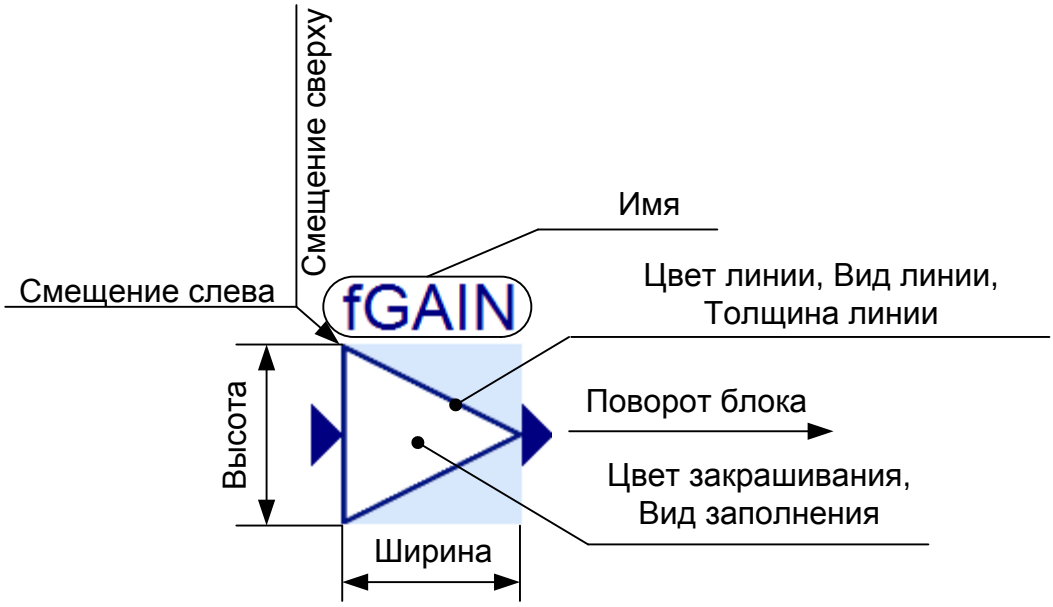


Рис. 22. Параметры блока

Имя блока, которое будет отображаться на поле набора.
Смещение слева и **Смещение сверху** координаты левого верхнего угла блока.
Ширина и **Высота** ширина и высота блока.

Настройка внешнего вида блока:
Цвет закрашивания и **Цвет линии** свойство цвета заливки и типа заливки.
Вид заполнения и **Вид линии** цвет и тип линии контура блока.
Толщина линии ширина контура блока.
Отображать имя показать или скрыть имя блока **.**
Направление поворот блока в одном из доступных направлений: Слева, справа, сверху, снизу.

Группа Исполнение

Внутренние параметры блока – параметры, участвующие в выполнении функции обработки данных, заложенных в блок. Подробней смотрите в документации к блоку.

Менеджер проекта

Менеджер представляет собой окно навигации, в котором, в виде вложенной структуры, представлен открытый проект.

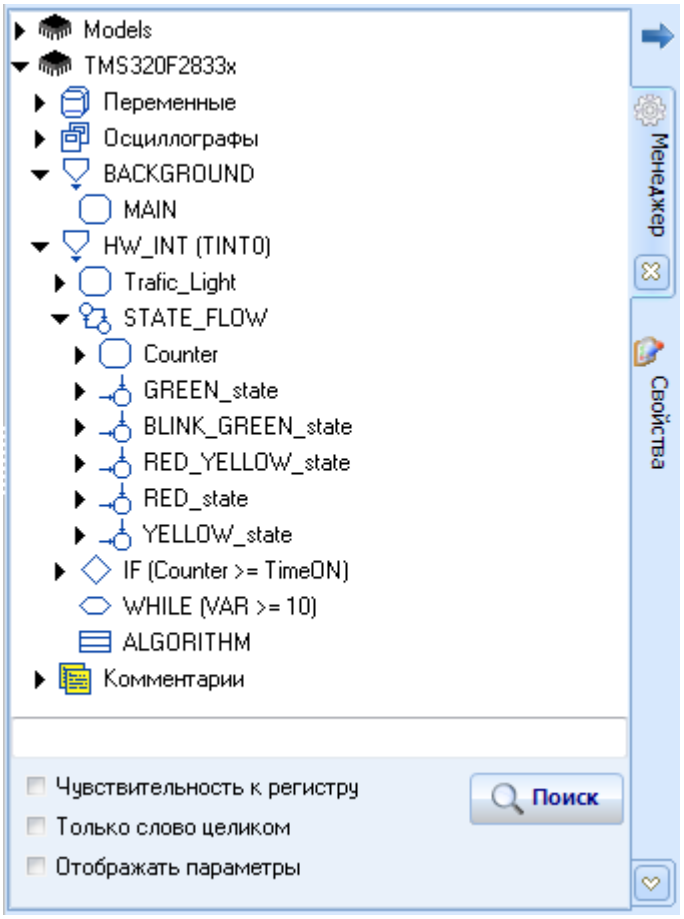


Рис. 23. Менеджер проекта Менеджер

Менеджер проекта содержит по дереву для каждой библиотеки.

В дерево библиотеки отображается только те блоки и системы, которые соединены с блоком **EVENT**. Не подключенные к EVENT блоки группируются в разделе Менеджера **Неиспользуемые блоки**. Также в пункт **Неиспользуемые блоки** добавляются блоки внутри ALGORITHM, которые не подключены к входу алгоритма.

Отдельным пунктом группируются **Переменные**. С помощью перетаскивания мышью можно задать желаемую очерёдность отображения переменных в дереве.

Для осциллографов, используемых в проекте, отдельно создаётся список, который можно использовать для перехода в схему с осциллографом.

Все пункты менеджера являются ссылками на соответствующие блоки. Если два раза нажать по пункту, то текущее положение в открытом проекте переключится к блоку.

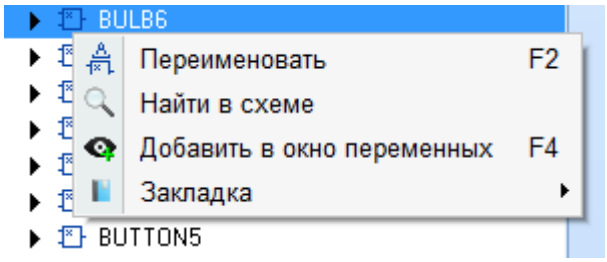


Рис. 24. Контекстное меню

В контекстном меню можно переименовать блок (**Переименовать**), найти на схеме блок (**Найти в схеме**), добавить блок в Окно переменных (**Добавить в окно переменных**), поставить закладку (**Закладка**).

Менеджер отображает порядок выполнения блоков. Перетаскивая курсором мыши можно изменить очерёдность выполнения блоков в программе.

Поиск по проекту

Окно менеджера проекта содержит пункт инструмент выполнения поиска по открытому проекту. Поиск происходит по имени блоков, которые находятся в проекте.

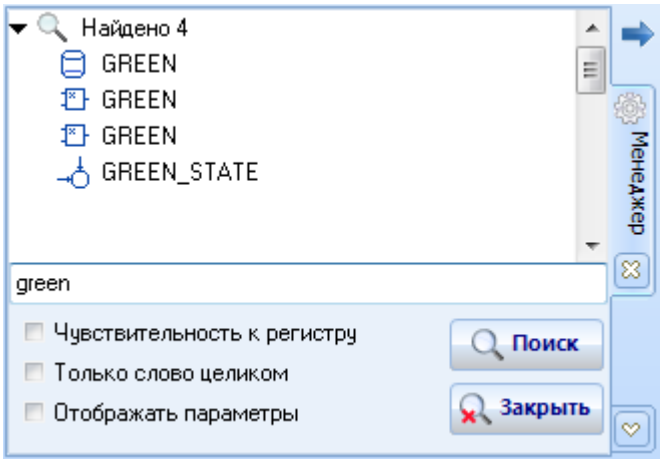


Рис. 25. Результаты поиска по проекту

Поиск можно осуществлять с параметрами:

- Чувствительность к регистру – поиск слов с разным регистром букв.
- Только слово целиком – поиск целого слова.
- Отображать параметры – поиск по параметрам внешних блоков.

Очистить окно поиска и перейти в менеджер проекта можно с помощью кнопки **Закрыть**.

Назначение закладок

На любой блок можно назначить закладку. Все закладки группируются в пункт **Закладки**. По умолчанию имя закладки состоит из пути к блоку. Можно переименовать закладку по своему усмотрению. Для перехода по закладке нужно дважды нажать ЛКМ по закладке.

Существует два пути назначения закладок.

Первый способ:

1. В **Менеджере** выбрать интересующий блок.
2. Нажать по нему ПКМ.
3. Выбрать пункт **Закладка**.

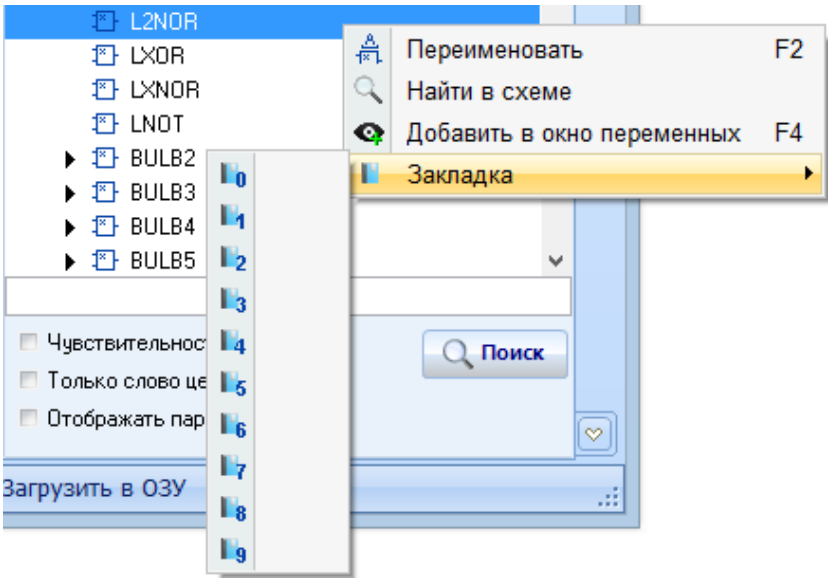


Рис. 26

4. Выбрать свободную закладку или перезаписать существующую.
5. В **Менеджере** появится группа **Закладки**.
6. Если удалить блок, на который назначена закладка – произойдёт удаление закладки из менеджера проекта.

Второй способ:

1. Перейти к блоку, которому будет назначена закладка.
2. Нажать ПКМ по блоку.
3. Выбрать пункт **Закладка**. Выбрать позицию для закладки.

К каждой закладке назначена горячая клавиша Ctrl+N, где N – цифры от 0 до 9. При нажатии Ctrl+0, произойдёт переход к месту программы, где назначена закладка 0 и т. п..

Шаблоны

Шаблоны предназначены для частичного либо полного сохранения схем для повторного использования через вкладку шаблоны.

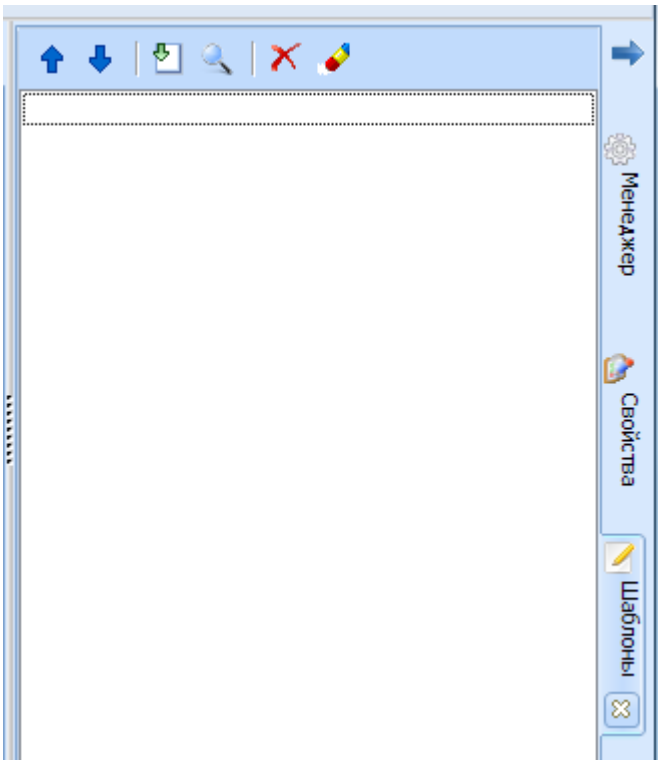


Рис. 27. Вкладка Шаблоны

Во вновь созданном проекте вкладка **Шаблоны** пуста. Для добавления готовых шаблонов необходимо нажать кнопку импортировать и выбрать файлы шаблонов.

	Перемещение выделенного шаблона по списку шаблонов.
	**Импортировать шаблоны** в проект. По умолчанию папка расположения шаблонов Templates в корне установленной программы.
	Поиск по списку шаблонов.
	**Удалить выделенный шаблон**. При удалении программа предложит удалить файл шаблона из списка или удалить из списка и из папки шаблонов (безвозвратно).
	**Очистить список шаблонов**

Для создания шаблона необходимо выделить схему на поле набора и ПКМ вызвать меню, в котором выбрать пункт Сохранить в шаблоны.

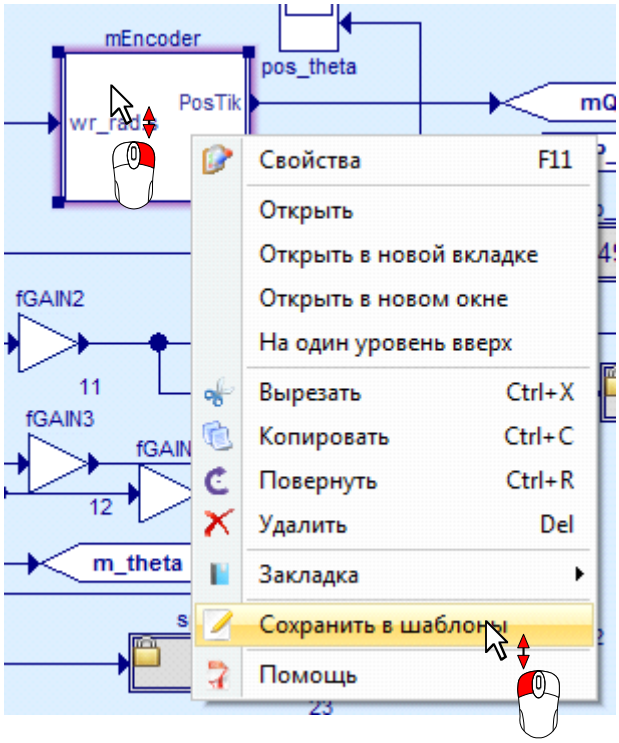


Рис. 28. Создание Шаблона

В появившемся окне задать имя и директорию сохранения шаблона.

Для добавления шаблона необходимо выделить нужный шаблон из списка Шаблонов и перетащить на поле набора.

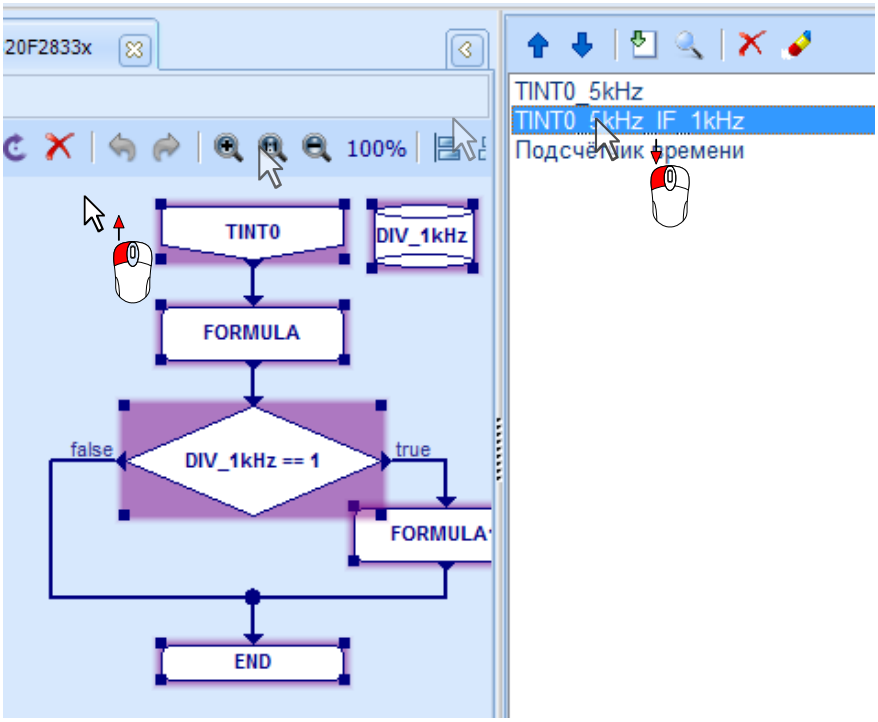


Рис. 29. Добавление шаблона на поле набора

Окно Переменные

Предназначено для отображения в процессе моделирования и работы с контроллером параметров блока, которые доступны для отображения и редактирования в реальном времени.

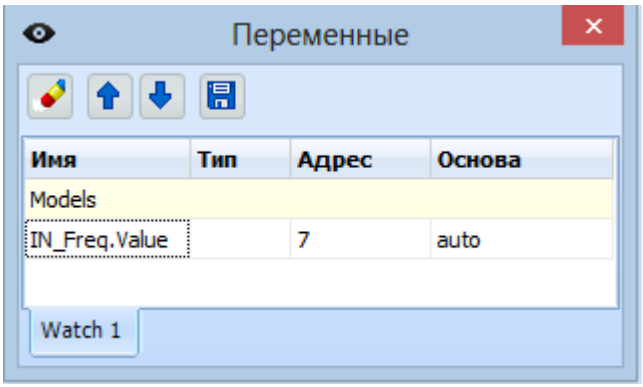


Рис. 30. Окно Переменные

1. Добавление элементов Переменные.

В окне **Переменные** можно добавить только параметры блока, и только те параметры, которые хранятся в буфере глобальных данных (тип переменных **Variable**).

1.1. Выделить нужные блоки.

1.2. Нажать клавишу **F4** или вызвать всплывающее меню нажатием правой кнопки мыши по одному из выделенных блоков и выбрать пункт меню **Добавить в окно переменных.** Также можно добавить в окно **Переменных** из **Менеджера** проекта.

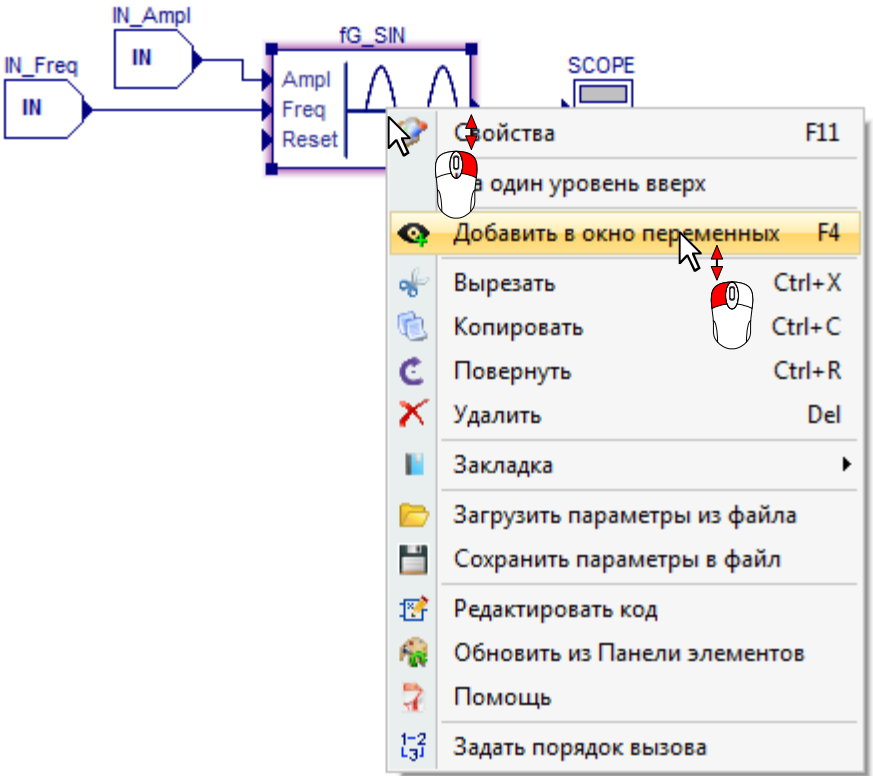


Рис. 31. Добавление в окно **Переменные** из контекстного меню

2. Удаление добавленных переменных из окна Переменные.

2.1. Необходимо ЛКМ нажать по имени переменной.

2.2. Нажать клавишу **Delete (Del)** – произойдёт удаление выделенных элементов окна **Переменные**.

2.3. Для того, чтобы очистить весь список необходимо нажать на кнопку с пиктограммой ..

3. Сохранение настроек параметров.

Нажатие кнопки сохраняет значения добавленных переменных параметров блока в **окно Переменных** в соответствующие параметры блоков. Таким образом, например, коэффициенты настроек регуляторов сохранятся в параметрах блока. Если не нажать сохранить, параметры при повторном запуске моделирования/обмена данными с чипом будут выставлены по умолчанию (как они заданы в окне **Свойства**).

4. Изменение порядка следования элементов окна Переменные.

С помощью кнопок выделенный элемент двигается вверх и вниз по списку элементов окна **Переменные** соответственно.

5. Изменение значения элемента окно Переменные.

Изменение значений элементов окна **Переменные** до запуска активного режима (моделирования/обмена данными с чипом) не приводит к реальным изменением настроек параметров.

6. Обновление

Для запуска получения данных в **окне Переменные** необходимо нажать кнопку **Начать моделирование** для режима моделирования, в режиме связи с чипом необходимо нажать **Начать обновление**.

7. Добавление новой вкладки

Для добавления вкладки необходимо нажать ПКМ по первой вкладке. Выбрать пункт **Добавить окно…**

Для редактирования имени вкладки выбрать **Изменить окно…**

Для удаления вкладки выбрать **Удалить окно…**

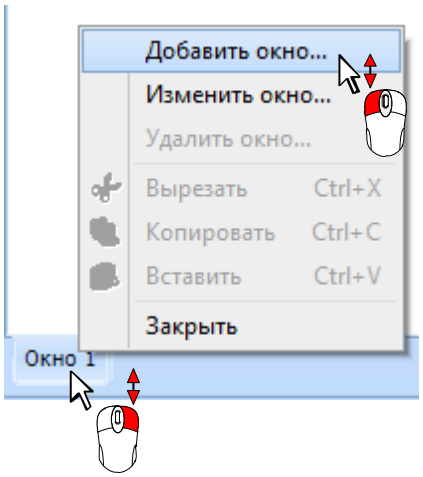


Рис. 32. Добавление новой вкладки в окно **Переменные**

Составление структуры программы из встроенных блоков

Структура программы состоит из встроенных блоков, расположенных в списке Embedded Панели элементов. Для создания экземпляра блока необходимо:

1. Захватить ЛКМ в палитре необходимый блок. Причём ЛКМ нужно нажать на название блока.
2. Удерживая ЛКМ с блоком, перенести на открытое поле набора.
3. Отпустить ЛКМ, на поле набора появится экземпляр блока.

Для соединения блоков между собой необходимо:

1. Нажать на треугольник выход блока. Появиться синяя соединительная линия, тянущаяся за курсором мыши.
2. Нажать на треугольник входа блока, к которому подключаем. Появится соединительная линия.
3. Можно прервать соединительную линию, нажав ПКМ по полю набора. Далее можно продолжить вести линию, нажав ЛКМ по кружку завершения линии.

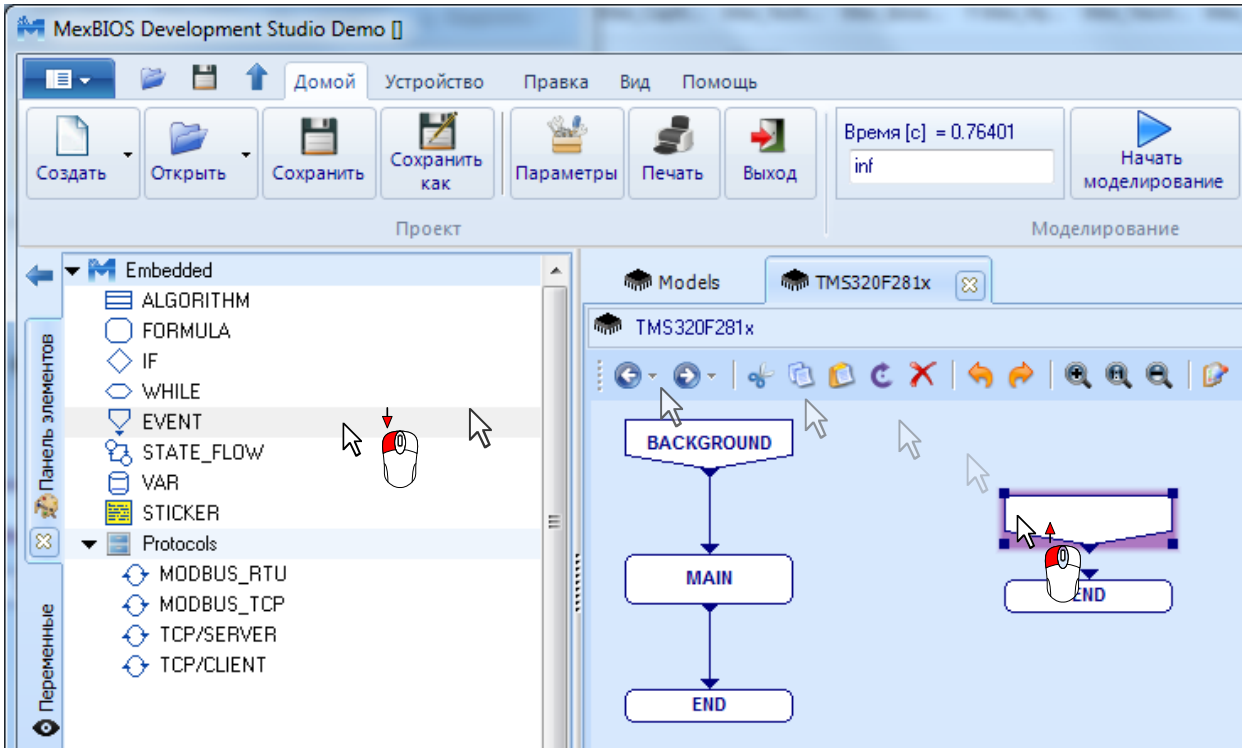


Рис. 33. Вынос встроенного блока на поле набора

Аналогичным образом выносятся другие блоки из списка **Embedded**. Смотрите пример построения программы в разделе «**Пример программы**».

Назначение встроенных блоков смотрите в разделе **Описание встроенных блоков**.

Если встроенные блоки (ALGORITHM, FORMULA, IF, WHILE, STATE_FLOW) не подключены к блоку EVENT (BACKGROUND или другое SW или HW прерывание), то они не исполняются в программе. Не подключенные внешние блоки отображаются в менеджере проекта в списке **Неиспользуемые блоки**.

Для выполнения периодичных задач необходимо создать периодическое прерывание (HW) с помощью блока EVENT. Периодическое прерывание настраивается в стартовом проекте, для каждой библиотеки имеется своё прерывание. Название прерывания смотрите в описании стартового проекта.

Составление и редактирование схем из внешних блоков

Внешние блоки из палитры можно добавить только в блоки FORMULA и STATE.

1. Создание экземпляра блока.

Создание экземпляра блока производится выносом блока из палитры на поле набора FORMULA или STATE.

1.1. Двумя кликами по FORMULA или STATE зайти внутрь блока. После этого произойдёт изменение палитры.

1.2. В палитре выбрать из раскрывающихся групп внешних блоков или из встроенных блоков необходимый элемент.

1.3. Захватить блок ЛКМ и, не отпуская, перенести элемент палитры на поле набора.

1.4. В точке отпускания левой кнопки мыши возникает новый экземпляр блока.

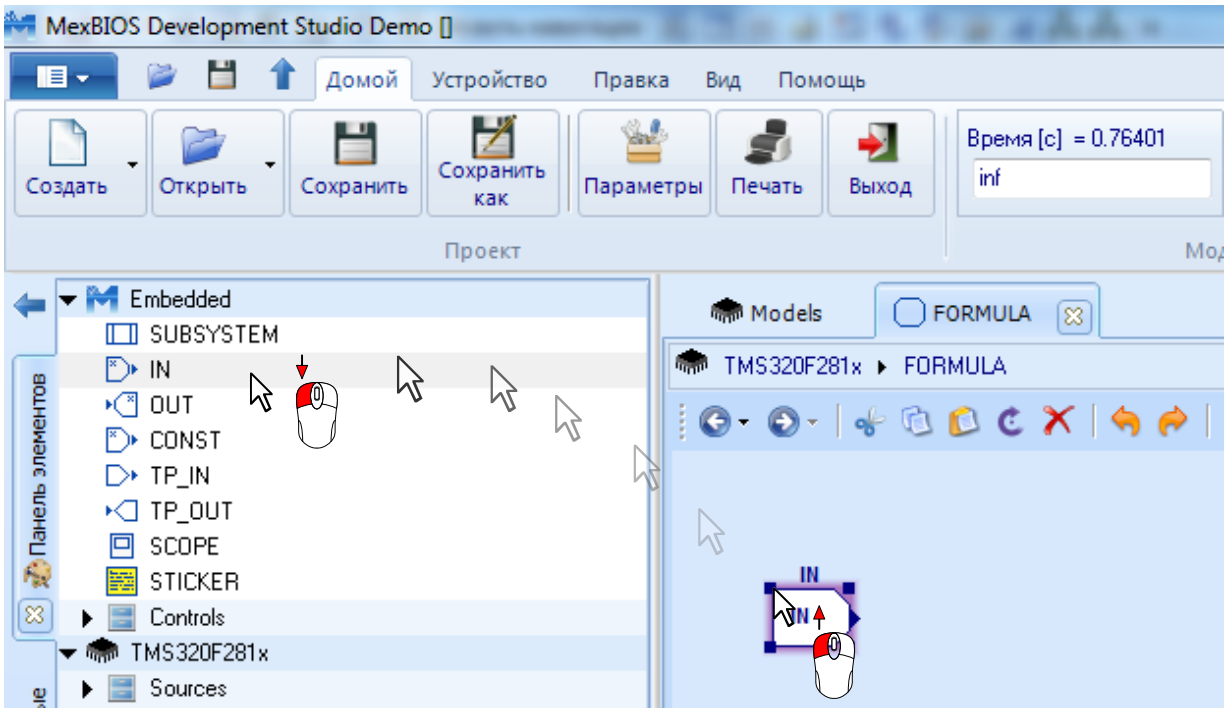


Рис. 34. Вынос блока IN из палитры на наборное поле

2. Выделение блоков.

Прежде чем начать выполнять манипуляции с любыми элементами схемы они должны быть выделены.

2.1. Одиночное выделение.

2.1.1. Выделение блока производится нажатием ЛКМ по блоку.

2.1.2. В углах выделенного блока появляются тяжки, потянув за которые, мышью можно изменить размер блока.

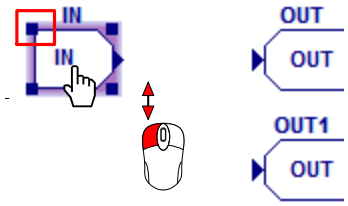


Рис. 35. Одиночное выделение

2.2. Выделение группы блоков.

2.2.1. Нажать левой кнопкой мыши на пустое место на поле.

2.2.2. Затем, не отпуская кнопку мыши, вести до тех пор, пока в возникшей рамке не окажутся все нужные блоки.

2.2.3. После того, как будет отпущена левая кнопка мыши, все элементы под рамкой окажутся выделенными.

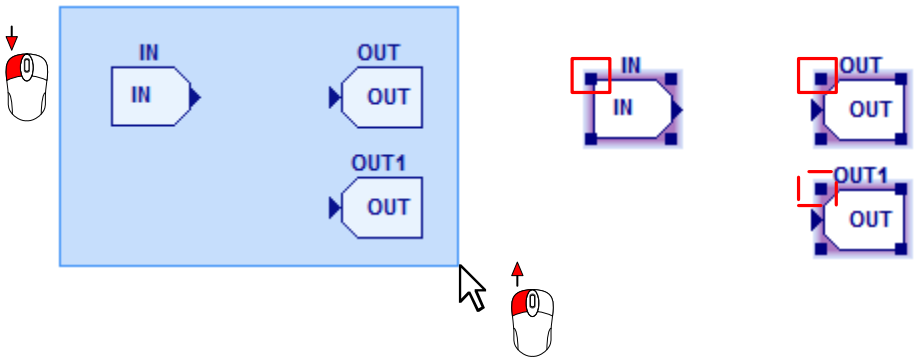


Рис. 36. Выделение группы блоков

2.3. **Последовательное выделение блоков с помощью клавиши Shift**.

2.3.1. Последовательное выделение выполняется так же, как и одиночное, но с нажатой клавишей **Shift**.

2.3.2. При этом клик с нажатой клавишей **Shift** на каждом не выделенном элементе будет добавлять его к группе выделенных элементов, а клик на уже выделенном элементе будет удалять его из группы выделенных элементов.

2.3.3. При групповом выделении рамкой с нажатием клавиши **Shift** блоки под рамкой добавляются к уже выделенным.

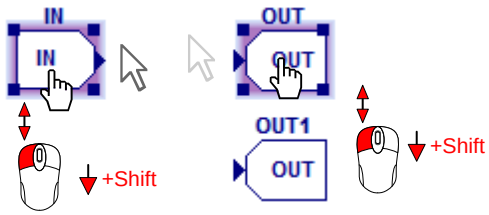


Рис. 37. Поочередное выделение с помощью клавиши Shift

2.4. **Снятие выделения.**

Однократное нажатие на пустом месте схемы снимает выделение со всех элементов сразу.

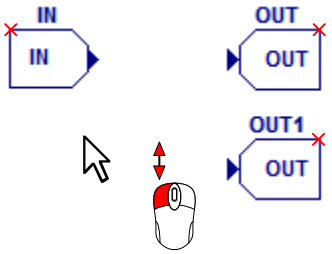


Рис. 38. Снятие выделения

3. Растягивание блока.

Растягивать можно только выделенные блоки.

3.1. Потянуть левой кнопкой мыши одну из тяжек выделенного блока.

3.2. После отпускания мыши блок примет новый размер.

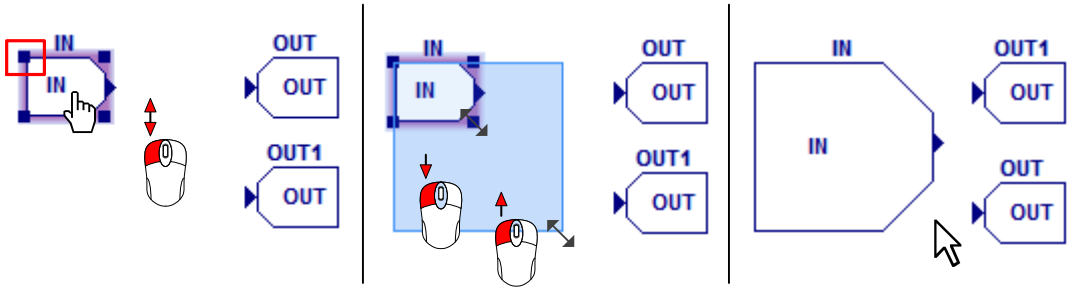


Рис. 39. Растягивание блока

4. Перенос блока (группы выделенных блоков).

Для переноса выделенного блока нужно нажать на выделенный блок и «перетащить» мышью в новое место.

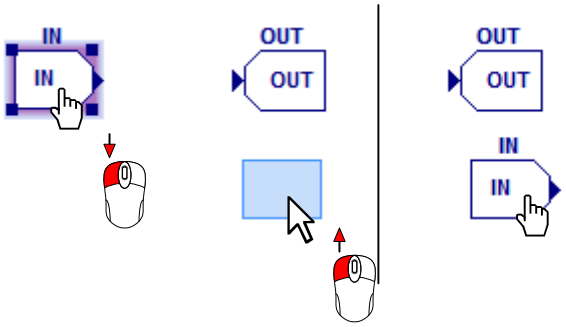


Рис. 40. Перенос блока

Аналогичным образом переносится группа выделенных блоков.

4.1. Нажать левой клавишей на один из выделенных блоков.

4.2. «Перетащить» всю группу в новое место.

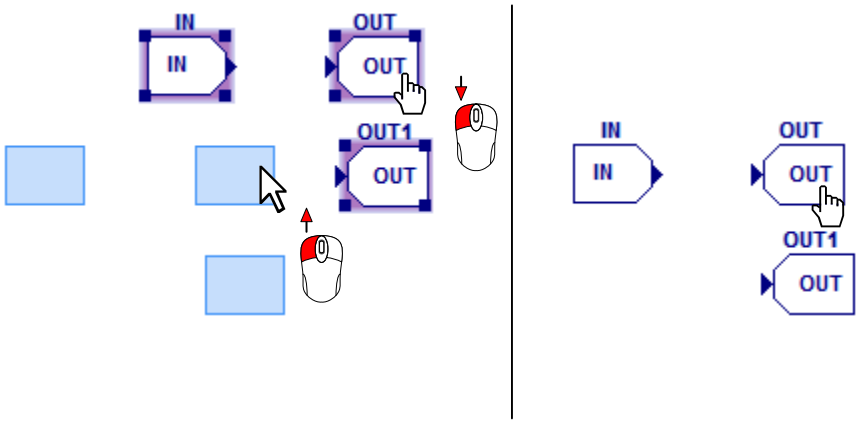


Рис. 41. Перенос группы блоков

5. Ручное конструирование линии.

Для создания соединительной линии можно нажатием на один из контактов блока начать процесс построения линии. После этого вслед за движениями курсора по полю, автоматически будет выстраиваться макет линии.

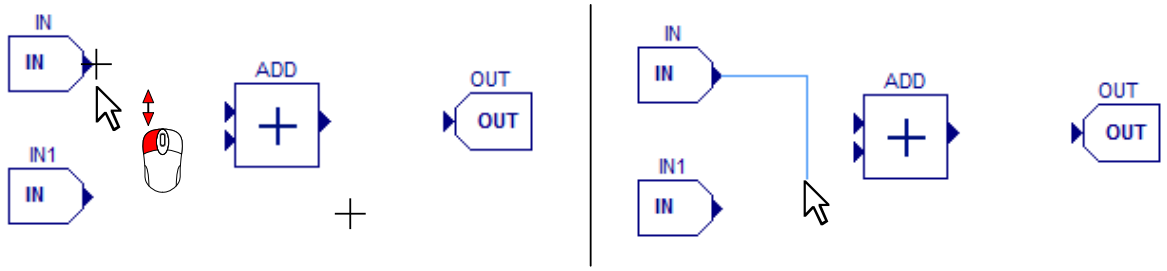


Рис. 42. Конструирование линии

6. Фиксация сконструированного участка линии.

Если в процессе конструирования линии сделать левый клик мышью, то конструируемый участок лини зафиксируется, и дальнейшее конструирование будет производиться от места клика.

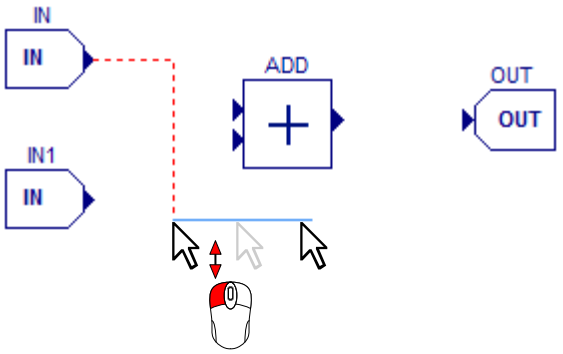


Рис. 43. Фиксирование построенного участка линии

7. Отмена конструирования линии (линия не завершена).

Если в процессе конструирования линии совершить правый клик мышью, то конструирование остановиться. При этом линия будет не завершенной.

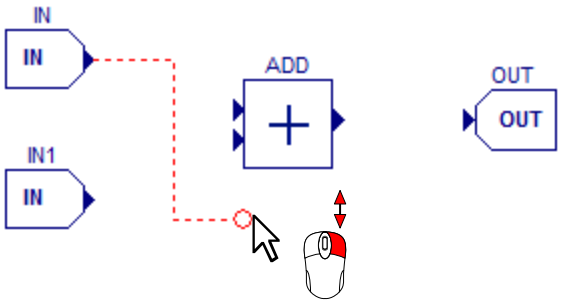


Рис. 44. Выход из режима конструирования линии

8. Завершение конструирования.

Для соединения с блоком нужно в процессе конструирования линии сделать левый клик по блоку, с которым планируется соединение. После этого появится завершенная линия.

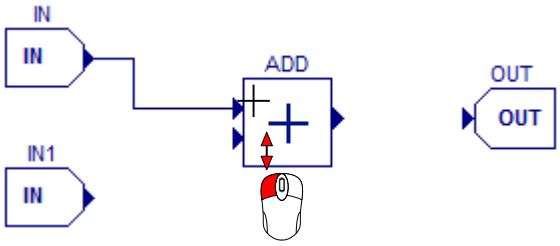


Рис. 45. Соединение с блоком

9. Автоматическое конструирование линии.

Для более быстрой и удобной работы в MexBIOS™Development Studio предусмотрен режим автоматического связывания блоков линиями. Для этого нужно:

9.1. Выделить один блок или группу блоков с выходными контактами.

9.2. Удерживая клавишу **Ctrl**, нажать на блок. MexBIOS™Development Studio автоматически соединит все выходные контакты выделенной группы с его входами.

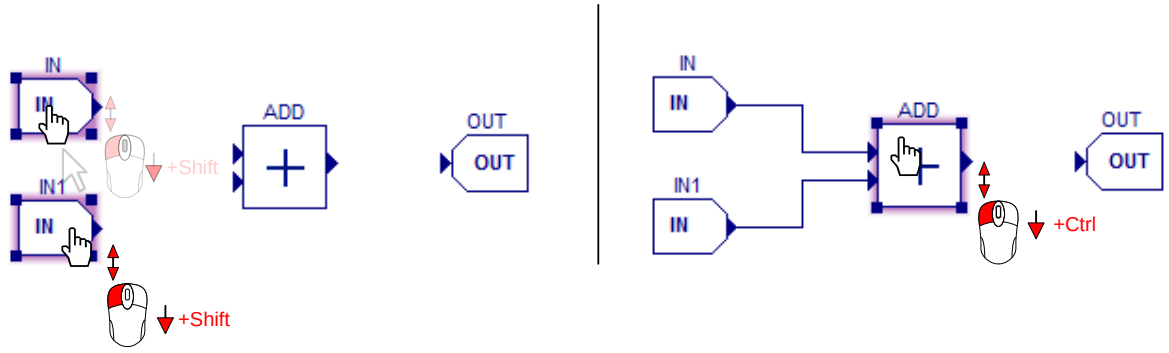


Рис. 46. Быстрое соединение

10. Продолжение конструирования незавершенной линии.

После завершения конструирования линии, и если линия не завершена (один или оба ее конца свободны), ее можно продолжить путём нажатия на свободный конец линии левой кнопкой мыши, что вновь возвращает режим конструирования линии.

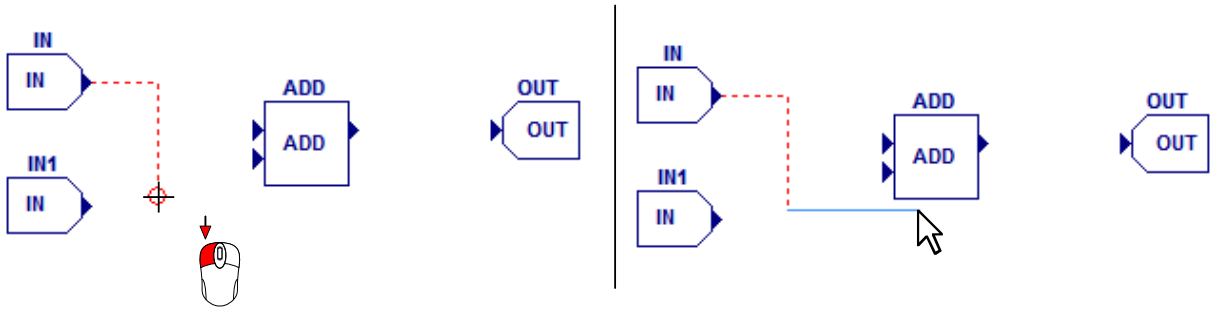


Рис. 47. Продолжение конструирования

11. Склеивание двух незавершенных линий.

Две незаконченные линии можно слить в одну законченную.

Для этого надо вновь вызвать режим конструирования для одной из линий и завершить конструирование кликом мыши по свободному концу другой. Тогда две линии превратятся в одну.

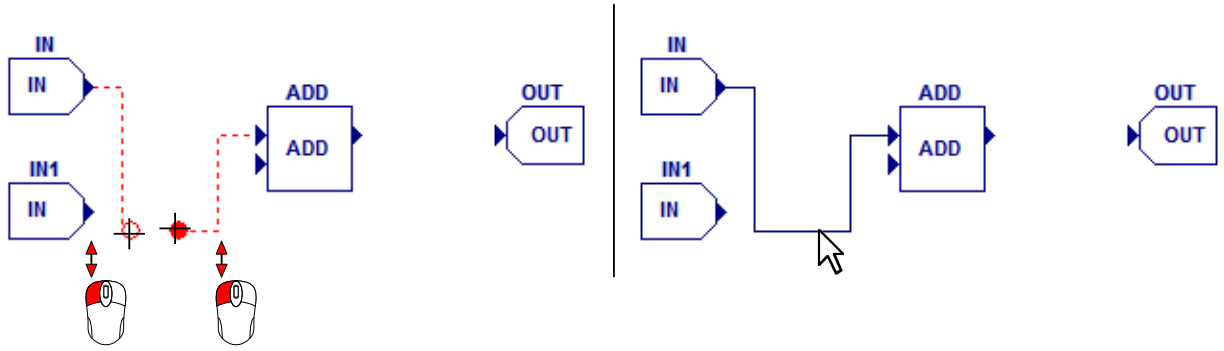


Рис. 48. Слияние линий

12. Вклеивание блока в линию связи.

При необходимости можно отдельно стоящий блок вклеить в уже существующую связь, просто перенеся блок на линию связи.

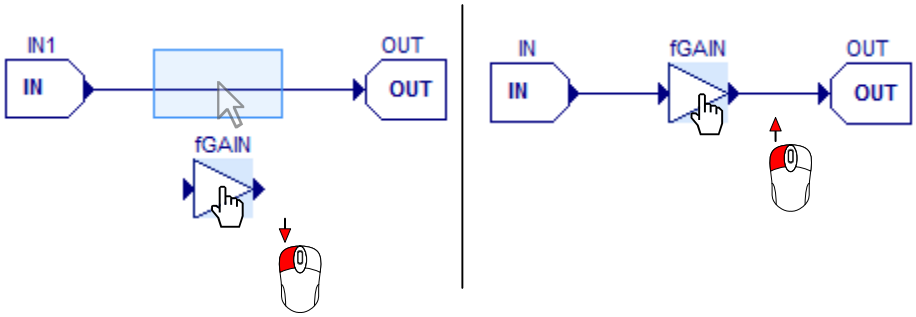


Рис. 49. Вклеивание блока в связь

13. Вырезание блока из линии связи.

Также можно блок вырезать из связи переносом блока с нажатой клавишей **Shift**. Действие обратное вклеиванию блока в линию связи. Необходимо захватить блок ЛКМ, перенести его на новое место, затем нажать клавишу Shift и отпустить ЛКМ.

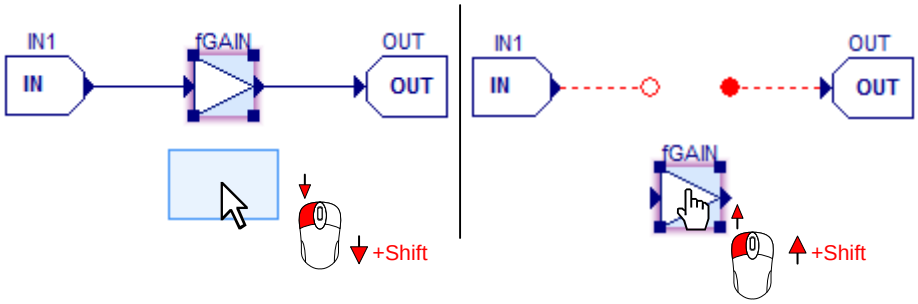


Рис. 50. Вырезание блока из связи

14. Вклеивание блока в разрыв.

Блок можно вклеить в разрыв между линиями, для этого нужно просто перенести блок в место разрыва.

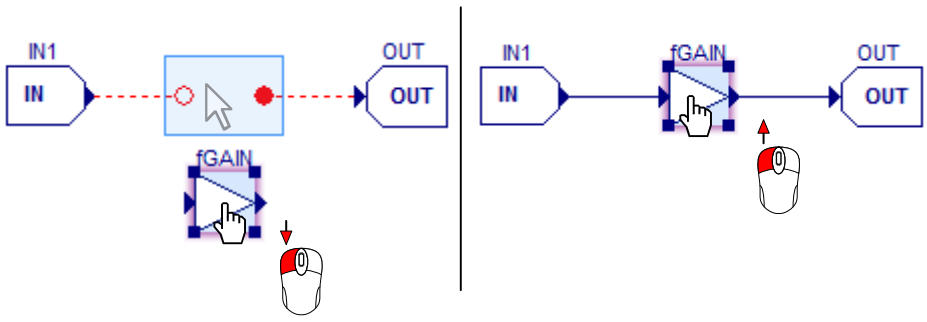


Рис. 51. Вклеивание блока в разрыв

15. Ветвление линии связи.

Линии связи можно разветвлять. Для этого нужно присоединить конструируемую линию к уже имеющейся связи. В месте соединения появится узел.

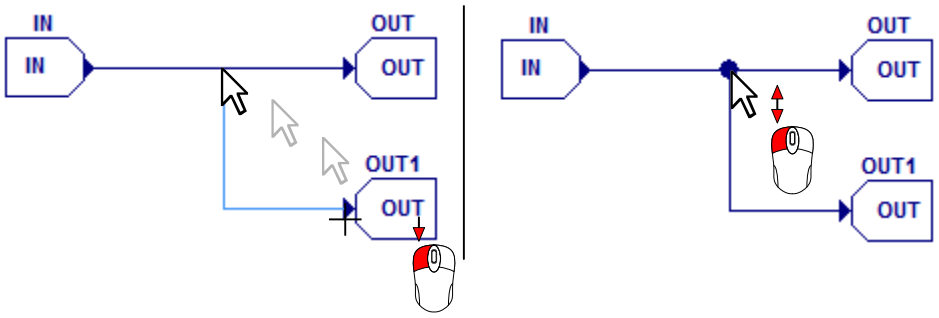


Рис. 52. Ветвление линий

Создать узел можно также нажатием правой кнопкой мыши по линии.

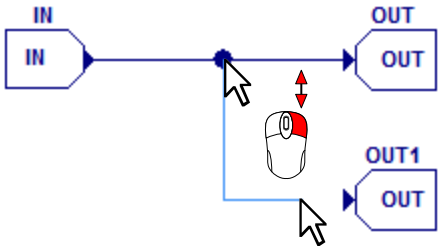


Рис. 53. Создание узла

16. Перенос узла ветвления линий связи.

При необходимости узел линий можно перенести. Для этого нужно нажать на узел левой кнопкой мыши и перетащить его на новое место.

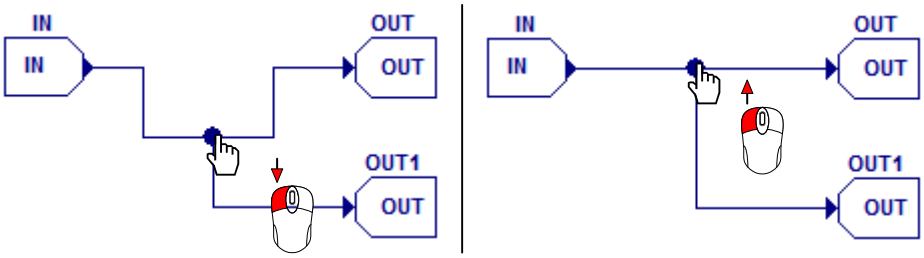


Рис. 54. Перенос узла

17. Перенос сегмента линии.

Также можно перенести сегмент линии.

17.1. Нажать левой клавишей мыши на линию.

17.2. Линия станет выделенной.

17.3. Не отпуская кнопку мыши, сместить сегмент линии в новое место.

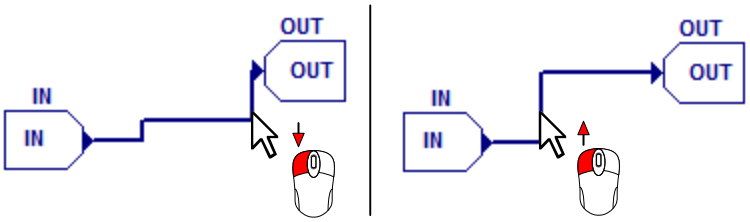


Рис. 55. Сдвиг сегмента

18. Удаление элементов схемы.

18.1. Нажатие кнопки **Delete** на клавиатуре удаляет выделенные элементы с поля набора.

18.2. Также выделенные элементы со схемы удаляет нажатие кнопки с соответствующей пиктограммой.

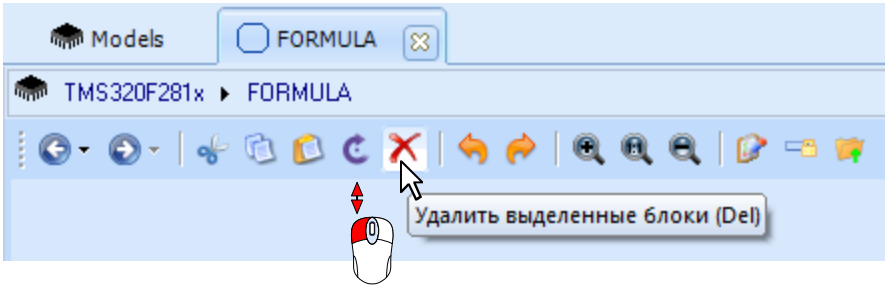


Рис. 56. Кнопка удаления

19. Вращение блоков.

19.1. Нажатие кнопки с соответствующей пиктограммой вращает выделенные блоки, поочередно меняя их направления: влево, вниз, вправо, вверх.

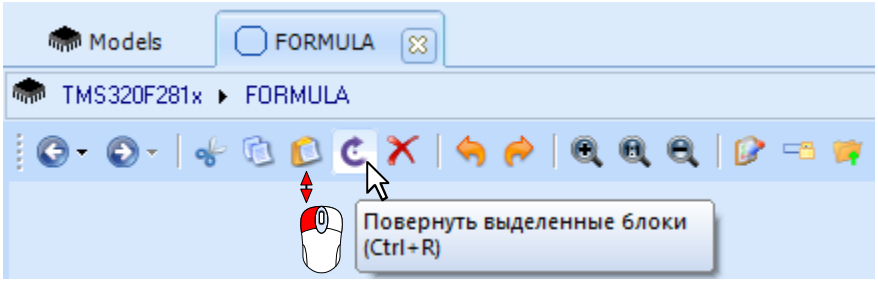


Рис. 57. Кнопка вращения

19.2. То же действие оказывает и сочетание клавиш **Ctrl+R**.

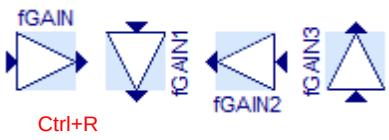


Рис. 58. Направления блока: влево, вниз, вправо, вверх

20. Выравнивание блоков по вертикали

20.1. Для выравнивания блоков по вертикали необходимо выделить группу блоков. И нажать кнопку из центральной панели, см. следующий рисунок:

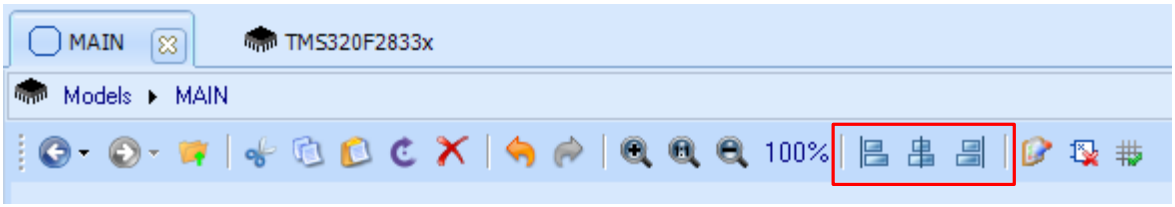


Рис. 59. Выравнивание блоков по вертикали

Выравнивание произойдёт по первому выделенному блоку.

20.2. При выделении группы блоков в панели **Свойства** отображаются общие свойства для выделенных блоков. Значения свойств отображаются для первого выделенного блока. При необходимости можно выделить **Смещение сверху** и нажать на поле набора ЛКМ – произойдёт выравнивание блоков по верхней координате.

21. Копирование элементов схемы

Копирование элементов производится нажатием соответствующих кнопок или привычным сочетанием клавиш для редактирования текста(смотрите раздел **Горячие клавиши**).

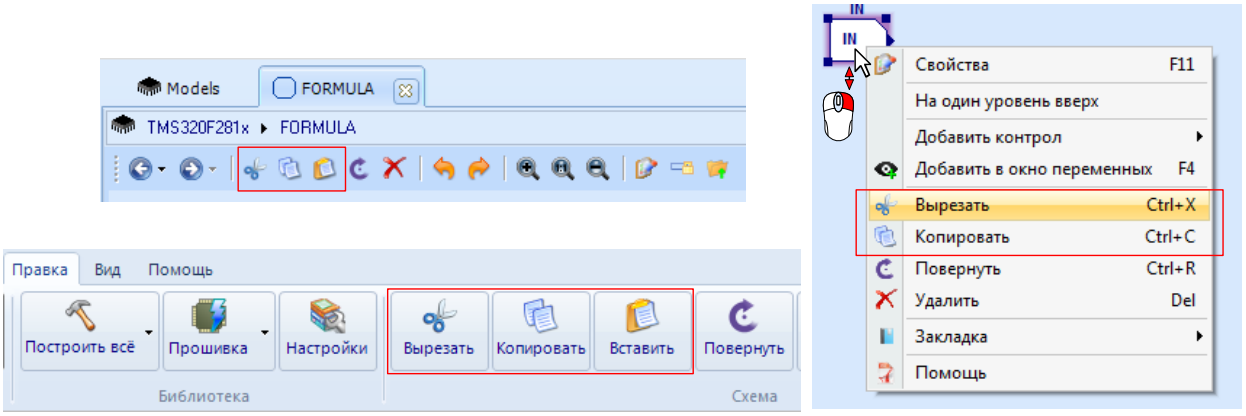


Рис. 60. Кнопки: вырезать, копировать, вставить

22. Копирование с помощью клавиши Ctrl

Если при переносе блока нажать клавишу **Ctrl**, то блок не перенесется, а скопируется в новое место.

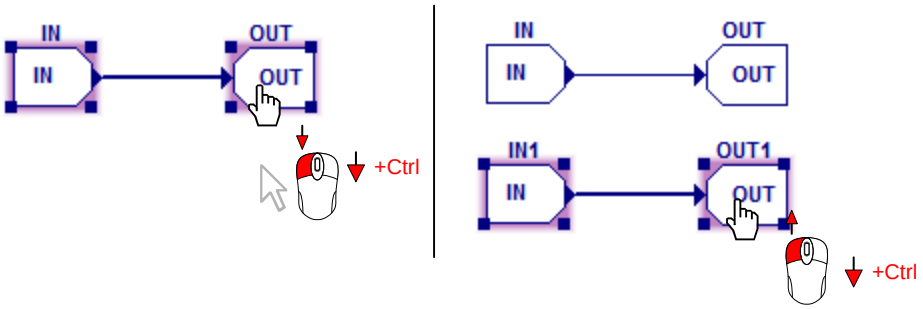


Рис. 61. Копирование с помощью клавиши Ctrl

Окно ошибок и предупреждений

В процессе работы с программой могут возникнуть ситуации, о которых необходимо знать пользователю. Или возникнуть ошибки.

Перечень ошибок и предупреждений приводит в нижней панели программы:

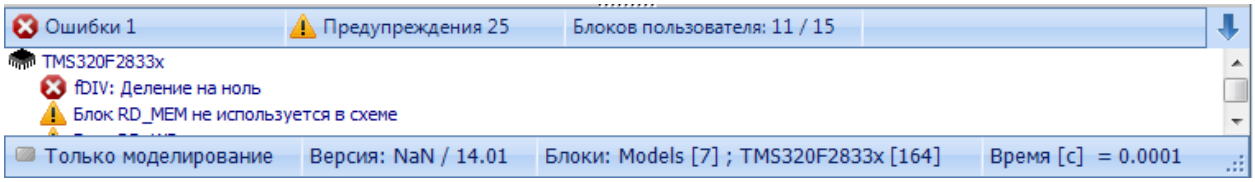


Рис. 62. Окно ошибок и предупреждений

Также в нижней панели приводится дополнительная информация:

Число блоков пользователя **Блоков пользователя 0/15** – означает, что пользователь не создал ни одного своего блока. В демонстрационной версии доступно создать 15 блоков.

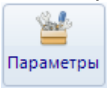
Только моделирование – показывает, что доступен режим моделирования. При подключении устройства будет отображаться информация об установленной связи.

Версии: Nan / 14.01 – первая цифра отображает версию ядра MexBIOS загруженной в контроллере, вторая цифра отображает версию ядра моделирования на компьютере.

Блоки: - отображает число использованных блоков в схеме.

Время – пройденное время моделирования в секундах.

Окно параметров

Для настройки параметров необходимо вызвать окно настройки (рис. 63), с помощью кнопки 

на панели управления **Домой**.

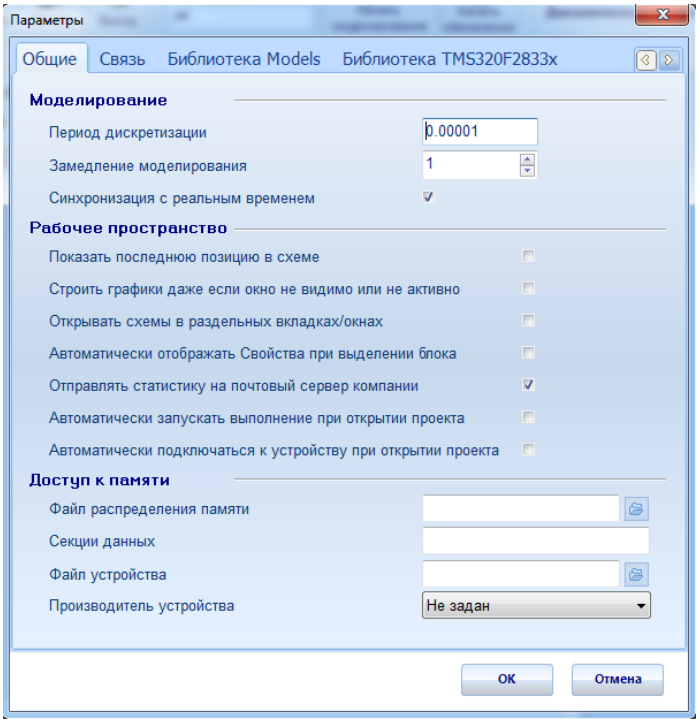
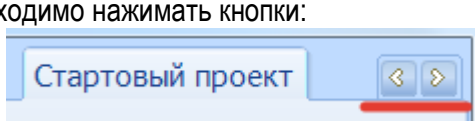


Рис. 63. Окно задания параметров

Окно содержит семь вкладок:

- **Общие** – содержит базовые настройки приложения, моделирования и проекта;
- **Связь** – настройки подключения к компьютеру платы с микроконтроллера;
- **Библиотека Models** – настройки для создания библиотеки блоков для Models;
- **Библиотека TMS320F2833x** – настройки библиотеки блоков для открытой библиотеки. На вкладке настраивается: распределение памяти, состав библиотеки, параметры компиляции. Смотрите раздел **Библиотека** (с. 133) ;
- **Стартовый проект** – настройки для компиляции стартового проекта. Смотрите раздел **Стартовый проект** (с. 138);
- **Утилиты для построения** – настройки компилятора библиотеки блоков. Смотрите раздел **Утилиты для построения** (с. 140);
- **Параметры справки** – настройки вызова пользователем справки.

Внимание: Не все вкладки помещаются в заголовок окна. Для переключения между скрытыми вкладками необходимо нажимать кнопки:



Вкладка Общие

Содержит следующие разделы:

- **Моделирование** – настройки для моделирования (далее - симуляции)

  Период дискретизации – определяет период дискретизации исполнения библиотеки моделей;

Внимание: Время **Период дискретизации** не может быть больше по величине времени, указанного в параметре **Период** блока **BACKGROUND** в схеме библиотеки блоков чипа. Если указать время меньше чем в параметре **Период**, то **Период дискретизации** автоматически станет равно параметру **Период**.

Задержка моделирования – степень замедления скорости моделирования;

Синхронизация с реальным временем – синхронизация симуляции с реальным временем. Можно использовать при несложных математических вычислениях.

- **Рабочее пространство** – настройки рабочего поля и приложения

Показывать последнюю позицию в схеме – при включенной настройке в рабочем поле отображается в виде красной метки последняя выбранная позиция;

Строить графики даже если окно не видимо или не активно – по умолчанию во время симуляции графики осциллографа (блока **SCOPE**) не отображаются, если его окно закрыто или свернуто (для уменьшения вычислительной нагрузки). Включение этой опции позволяет построение графиков во всех режимах.

Открывать схемы в раздельных вкладках/окнах – некоторые блоки содержат внутри себя отдельную подсхему для добавления в них блоков (**ALGORITHM**, **FORMULA** и т.д.) и при их открытии новая вкладка/окно не создаются, а используется тоже самое рабочее поле. Включение этой опции приводит к тому, что все подсхемы создаются в новой вкладке или окне.

Автоматически отображать свойства при выделении блока – по умолчанию при выделении любого блока не отображается окно инспектора (для просмотра и редактирования его параметров) и возможно вызвать лишь нажатием клавиши F11 или через меню «Вид – Свойства». При включении этой опции данное окно отображается сразу при выборе блока.

Отправлять статистику на почтовый сервер компании – при включенной опции при закрытии приложения отправляется статистическая информация использования приложения и библиотек на почтовый сервер компании.

Автоматически запускать выполнение при открытии проекта – при включенной опции при открытии любого проекта среда автоматически запускает расчет проекта.

Автосохранение проекта – Автоматическое сохранение проекта каждые 5 минут.

Автоматически подключаться к устройству при открытии проекта – при включении данной опции при открытии любого проекта среда автоматически подключается к коконтроллеру, если программа собрана в библиотеке микроконтроллера.

- **Доступ к памяти** – содержит опции подключения дополнительных файлов для более удобного использования блоков прямого доступа к памяти (**RD_MEM** и **WR_MEM**)

Внимание: Данный раздел не отображается, если проект только для библиотеки **Models**.

Если пользователь разрабатывает собственный проект программы и имеет значительные наработки (функции, структуры и т. д.), то существует возможность включить ядро MexBIOS в проект пользователя и через среду **MexBIOS™ Development Studio** обратиться к переменным и структурам проекта пользователя с помощью блоков **RD_MEM** и **WR_MEM**. Данные блоки обращаются непосредственно по заданному физическому адресу в память для чтения и записи в указанную ячейку памяти информации.

По умолчанию, при использовании блоков прямого доступа к памяти (**RD_MEM** и **WR_MEM**) для корректного отображения и задания значений переменных, необходимо задавать его физический адрес, смещение в памяти и размер. Для упрощения процесса определения адресов требуемых ячеек памяти (фактически переменных стартового проекта) пользователь должен задать следующие параметры:

Файл распределения памяти – определяет файл распределения памяти, данный файл обычно формируется при компиляции;

Секции данных– перечисленные через символ «;» секции памяти, в которых находятся параметры, структуры и т.п. (например, «L0SARAM;H0SARAM;ADC»)

Файл устройства – дополнительный файл описания параметров структур (подэлементы структур не определяются в файле распределения памяти). В общем случае, когда выбрано в параметре **Производитель устройства** (описание параметра смотри ниже) выбрано значение «Default», данный файл должен являться текстовым файлом, в котором каждая строка отражает настройку для отдельного параметра и задается в следующем формате:

Name**[\t]**Type**[\t]**Offset**[\t]**FriendlyName

Здесь	Name	– имя параметра или структуры из фала распределения памяти;
	Type	– отражает его размер (0-short или 1 слово, 1-long или 2 слова);
	Offset	– смещение в памяти;
	FriendlyName	– дружественное имя, отображаемое в инспекторе для параметра;
	[\t]	– символ табуляции.

Например, если структура с именем «myStruct» содержит следующие элементы: «long a, short b, short c», то для обращения ко всем ее параметрам в конфигурационном фале должны быть следующие строчки:

myStruct**[\t]**1**[\t]**0**[\t]**MyStruct.a
myStruct**[\t]**0**[\t]**2**[\t]**MyStruct.b
myStruct**[\t]**0**[\t]**3**[\t]**MyStruct.c

Производитель устройства – производитель дополнительного файла описания структур,

определяющий тип хранимой в нем информации. Возможны следующие значения:

Не задан – тип файла не определен (то есть файл не используется)

По умолчанию – текстовый файл в формате, описание которого представлено выше

В качестве примера приведено настройки на панели «**Свойства**» для обращения к элементам структуры «strctur» без использования **Файла устройства**.

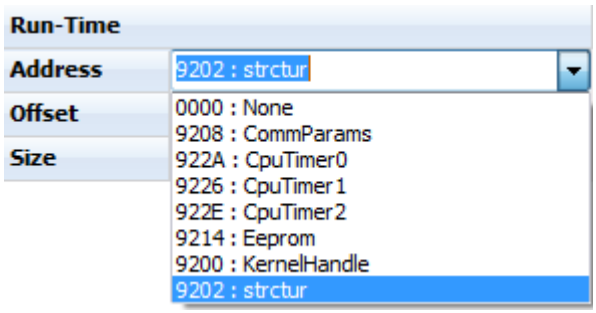


Рис. 64. Список доступных переменных стартового проекта из секции L1SARAM

В добавленном блоке **RD_MEM** выбрать адрес переменной или структуры, к которой будет привязан блок. Параметр Address.

В пункте Offset указать смещение, если обращение идёт к структуре. Пример вычисления нужного смещения показан на рисунке:

```
typedef struct {   //Offset
short    a;        //0
int      b;        //1
long     c;        //2
int      d ;       //4
} STRCT;
```

Рис. 65. Назначение смещения при обращении к структуре

Для значений **int**, **short** смещение равно 1 (начиная с нулевого смещения). Для значений формата **long** смещение равно 2, так как число занимает два слова в памяти контроллера.

Создадим структуру и запишем в тексте программы некоторые значения:

```
strctur.a=17;
strctur.b=13;
strctur.c=155;
strctur.d=7;
```

Параметр Size указывает длину слова. **Short** – одно слово, **Long** – два слова.

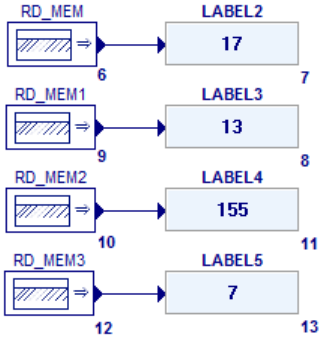


Рис. 66. Блоки **RD_MEM**

Вкладка Связь – настройка подключения

На вкладке содержатся опции для настройки связи программы с аппаратным обеспечением с предустановленным **MexBIOS™ kernel**.

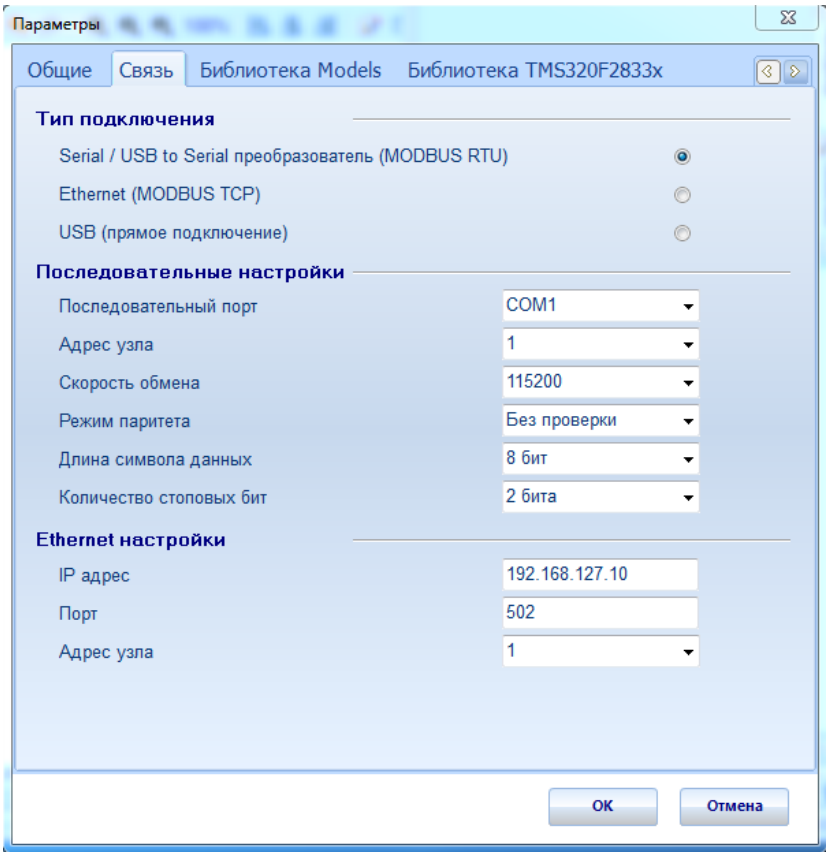


Рис. 67. Окно настройки параметров подключения

Содержит следующие разделы:

- **Тип подключения** – тип интерфейса подключения. Поддерживаются следующие последовательные интерфейсы: RS-232, RS-485 по протоколу MODBUS RTU; Ethernet с протоколом MODBUS TCP и встроенный интерфейс USB. Для организации связи в стартовом проекте должен быть подключен соответствующий драйвер.
- **Последовательные настройки** – настройки подключения по протоколу MODBUS RTU

Последовательный порт – порт, к которому подключено аппаратное обеспечение.
Адрес узла, Режим паритета, Длина символа данных, Количество стоповых бит – дополнительные параметры настройки протокола.
Скорость обмена – скорость коммуникации. Параметр должен соответствовать значению, заданному в стартовом проекте.

- **Ethernet настройки** – настройки подключения по протоколу MODBUS TCP

IP адрес – адрес устройства**.**
Порт – номер порта.
Адрес узла – номер подчинённого устройства.

- Для USB дополнительных настроек не предусмотрено.

Установка связи между компьютером и чипом

Создание подключения

В главном меню перейти на вкладку **Связь**. Если порт связи не выбран, то необходимо выбрать порт связи и протокол, по которому будет происходить обмен данными между компьютером и аппаратным обеспечением, в настройках **Параметры** (**см. предыдущий пункт**).

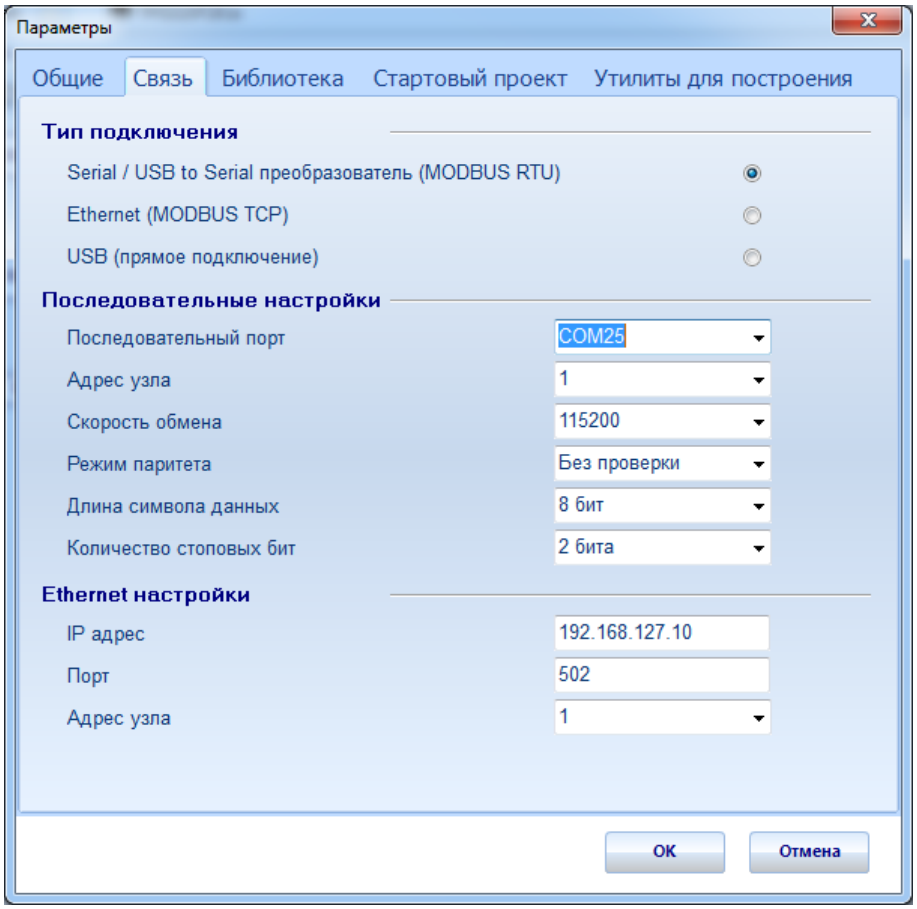


Рис. 68. Настройка связи

В разделе **Последовательные настройки**, в пункте **Последовательный порт** выбрать COM порт, к которому подключено аппаратное обеспечение. Если нет особых настроек связи, нажать кнопку OK.

Чтобы установить связь с чипом нажмите на кнопку **Подключиться**. Если подключение произошло удачно, то пиктограмма и подпись кнопки сменится. Чтобы разорвать связь с чипом, необходимо повторно нажать на кнопку **Отключиться**.

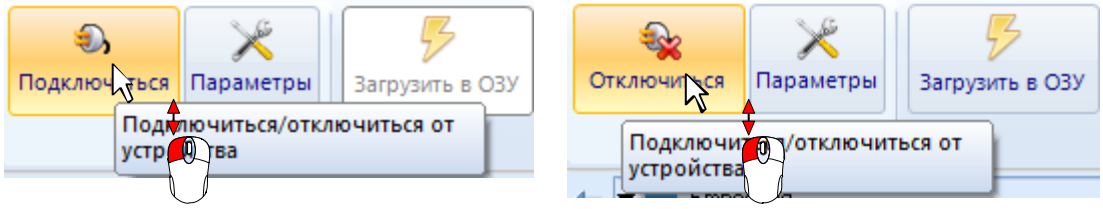


Рис. 69. Установка и разрыв связи с чипом

Если **COM** порт указан неверно или нет подключения, то появится сообщение об ошибке: **Порт для связи не доступен или уже используется.**

Вкладка Параметры справки

На вкладке **Параметры справки** можно настроить источник загрузки справки. Можно использовать справку в Интернете и локальную справку.

Для использования локальной справки необходимо скачать и установить набор pdf документов с сайта разработчика программы и переключить на соответствующий параметр на вкладке **Параметры справки**.

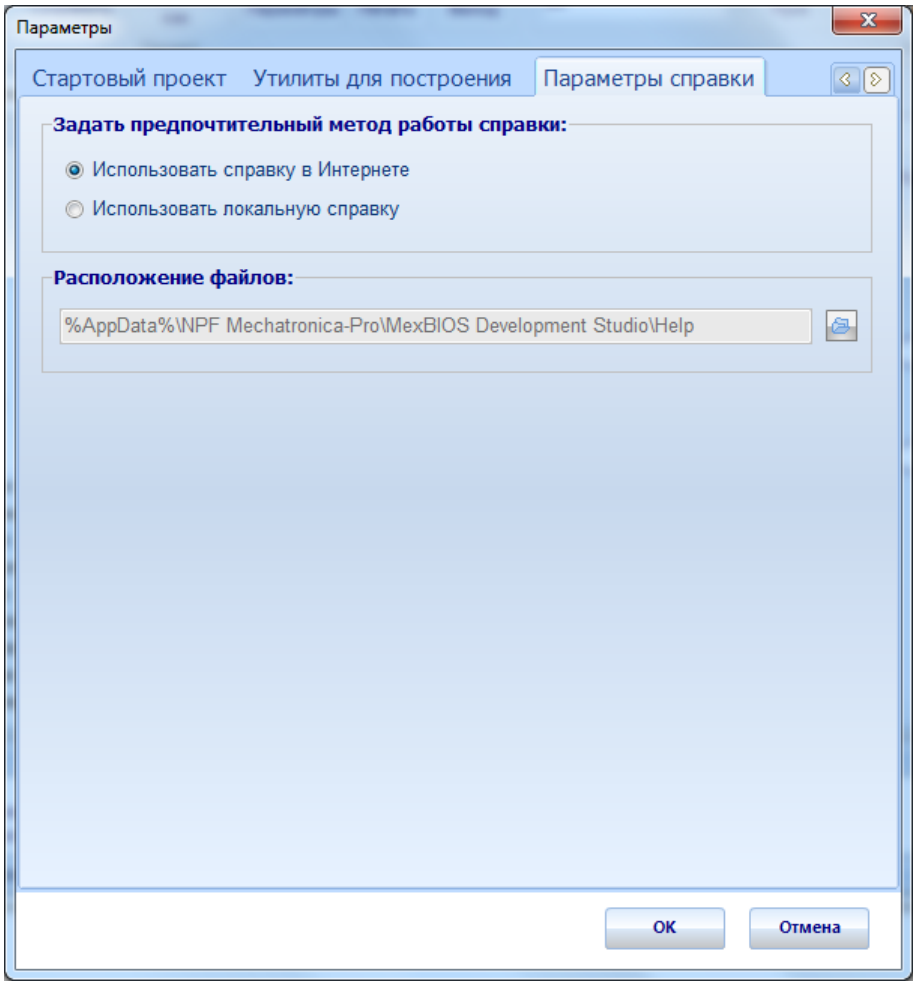
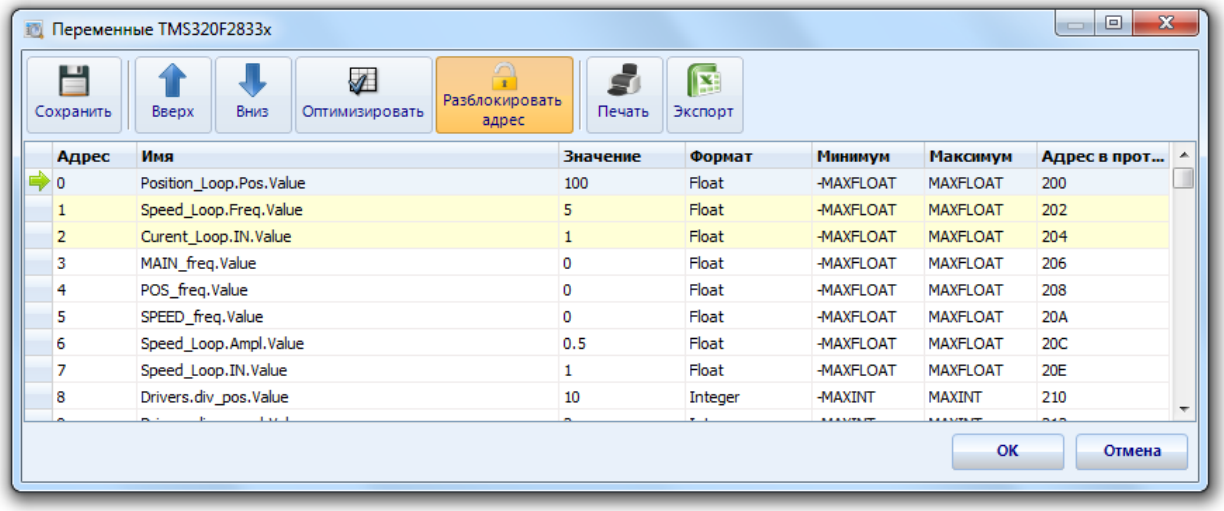


Рис. 70. Окно Параметры

Окно Таблица переменных

Окно **Таблица переменных** предназначено для просмотра и оптимизации параметров/переменных блоков, передаваемых в процессе обмена между **MexBIOS™ Development Studio** и контроллером по протоколу ModBUS RTU. Адреса (**Таблица переменных)** параметров можно использовать для управления устройством с программой, написанной в **MexBIOS™ Development Studio,** от стандартных внешних устройств, поддерживающих протокол ModBUS RTU.

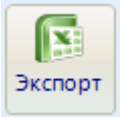


Адрес	Имя	Значение	Формат	Минимум	Максимум	Адрес в прот...
0	Position_Loop.Pos.Value	100	Float	-MAXFLOAT	MAXFLOAT	200
1	Speed_Loop.Freq.Value	5	Float	-MAXFLOAT	MAXFLOAT	202
2	Curent_Loop.IN.Value	1	Float	-MAXFLOAT	MAXFLOAT	204
3	MAIN_freq.Value	0	Float	-MAXFLOAT	MAXFLOAT	206
4	POS_freq.Value	0	Float	-MAXFLOAT	MAXFLOAT	208
5	SPEED_freq.Value	0	Float	-MAXFLOAT	MAXFLOAT	20A
6	Speed_Loop.Ampl.Value	0.5	Float	-MAXFLOAT	MAXFLOAT	20C
7	Speed_Loop.IN.Value	1	Float	-MAXFLOAT	MAXFLOAT	20E
8	Drivers.div_pos.Value	10	Integer	-MAXINT	MAXINT	210

Рис. 71. Окно Таблица переменных

Назначение кнопок:

Сохранить	Сохранение произведённых изменений. Для сохранения настроек также можно нажать кнопку OK. Диалог Parameters при этом закроется.
Вверх Вниз	Передвинуть **Вверх** или **Вниз** выделенную строку.
Оптимизировать	Оптимизировать адресное пространство протокола связи. После нажатия на кнопку производится удаление зарезервированных адресов.
Заблокировать адрес	Заблокировать выделенный адрес. После блокировки адрес не может быть изменён при оптимизации. Заблокированный адрес может быть разблокирован.
Печать	Напечатать таблицу данных с помощью принтера установленного по умолчанию.

	Экспортировать таблицу в формате *.xls

Назначение колонок:

Адрес - адрес в схеме проекта. Показывает очередь опроса и отображения данных на графических элементах схемы и в окне **Переменные**.

Имя - имя переменной, включающее название блока и формулу в котором находится блок.

Значение – значение переменной.

Формат – формат данных.

Минимум и **Максимум** – максимальное и минимальное возможное значение параметра.

Адрес в протоколе – адрес переменной в протоколе обмена данных.

Выделение адресов происходит автоматически для каждого нового блока, вынесенного на поле набора (для назначения адреса, тип параметра блока должен быть **Variable**). В процессе создания схемы, могут возникнуть пробелы в адресном пространстве из-за удаления блоков. Такие пробелы отображаются строкой **Зарезервировано** в окне параметров. Наличие пустых – зарезервированных адресов замедляет процесс обмена данными. Для оптимизации адресного пространства необходимо нажать кнопку **Оптимизировать** и нажать кнопку OK.

Адреса можно группировать и передвигать с помощью кнопок **Вверх** и **Вниз**. Процесс обновления данных будет происходить быстрее, если адреса будут сгруппированы по формулам, в которых происходит обращение по данным адресам.

Блоки организации коммуникации

Для организации коммуникаций между MexBIOS™ Development Studio и любым устройством, поддерживающим протокол MODBUS_RTU, TCP/MODBUS_RTU, TCP существуют специальные блоки **Embedded→Protocols**. Возможно организовать обмен данными между программой, созданной в MexBIOS™ Development Studio и сторонним устройством по протоколам **MODBUS_RTU**, **TCP/MODBUS_RTU**, **TCP**.

Подробней о протоколе **ModBUS** смотрите на сайте разработчика **Modbus** (http://www.modbus.org/).

Протокол MODBUS_RTU

На главное поле набора необходимо вынести блок **Embedded→Protocols→ MODBUS_RTU.**

Настроить параметры блока в окне **Свойства**:

Порт – выбор последовательного порта.

Скорость обмена – скорость обмена данными.

Паритет – режим паритета.

Длина данных – размер передаваемого символа.

Количество стоповых бит – количество стоповых бит.

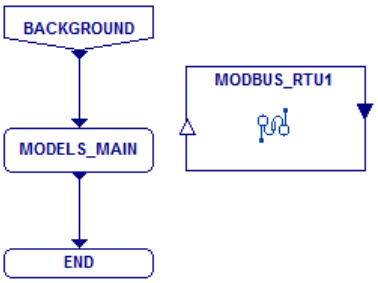


Рис. 72. Блок **MODBUS_RTU1**

Для обращения к данным, передаваемым по протоколу на поле набора в FORMULA необходимо вытащить блок **Embedded→Controls→PROTO_IN** для считывания данных и **PROTO_OUT** для записи данных**.** При выносе на поле набора появится список, из которого необходимо выбрать требуемый протокол. Таким образом определиться привязка блока к протоколу. Если на поле набора не добавлен один из блоков **Protocols**, то блоки **PROTO_IN** и **PROTO_OUT** не добавятся на поле набора.

В настройках **PROTO_IN**:

Адрес узла – адрес подчинённого устройства (адрес узла);

Функция – функция по которой будет обращаться данный блок по протоколу;

Адрес – адрес, по которому будет производиться считывание данных.

Количество – количество считываемых слов;

В настройках **PROTO_OUT**:

Адрес узла – адрес подчинённого устройства (адрес узла);

Функция – функция, по которой будет обращаться данный блок по протоколу;

Адрес – адрес, по которому будет производиться запись данных.

Количество – количество считываемых слов;

Протокол MODBUS_TCP

На главное поле набора необходимо вынести блок **Embedded→Protocols→ MODBUS_TCP.**

Настроить параметры блока в окне **Свойства**:

Host – указать IP адрес сервера.

Port – указать порт.

Подробней о протоколе **ModBUS** смотрите на сайте разработчика **Modbus**.

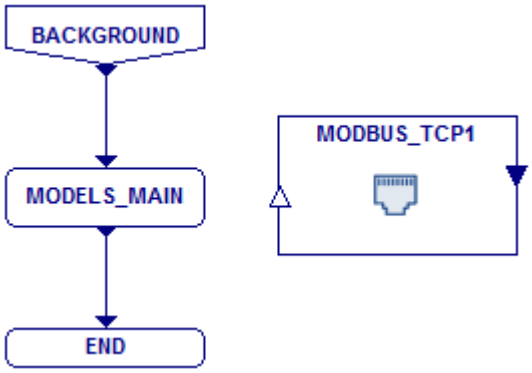


Рис. 73. Блок **MODBUS_TCP**

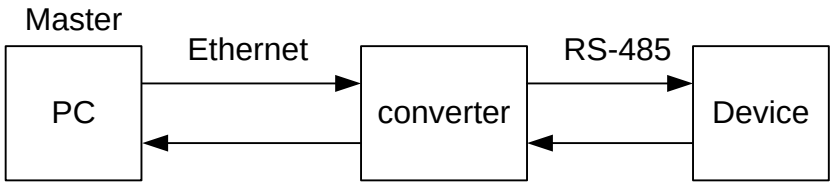


Рис. 74. Схема соединения блока MODBUS_TCP при использовании преобразователя (например MOXA, настроенной в режим TCP-Server)

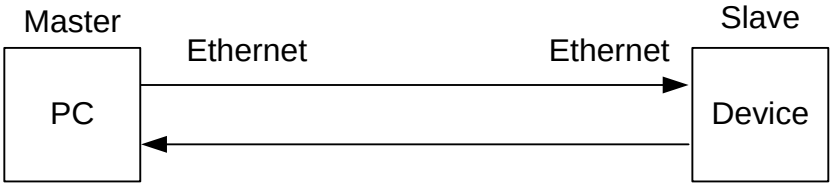


Рис. 75 Схема соединения блока MODBUS_TCP к устройству с Ethernet контроллером

Для обращения к данным, передаваемым по протоколу на поле набора в FORMULA необходимо вытащить блок **Embedded→Controls→PROTO_IN** для считывания данных и **PROTO_OUT** для записи данных**.** При выносе на поле набора появится список, из которого необходимо выбрать требуемый протокол. Таким образом определиться привязка блока к протоколу.

В настройках **PROTO_IN**:

Адрес узла – адрес подчинённого устройства (адрес узла);

Функция – функция по которой будет обращаться данный блок по протоколу;

Адрес – адрес по которому будет производиться считывание данных.

Количество – количество считываемых слов;

В настройках **PROTO_OUT**:

Адрес узла – адрес подчинённого устройства (адрес узла);
Функция – функция по которой будет обращаться данный блок по протоколу;
Адрес – адрес по которому будет производиться запись данных.
Количество – количество считываемых слов;

TCP/IP

Поддержка протокола связи **TCP/IP** представлена двумя блоками:
TCPServer и **TCPClient**.

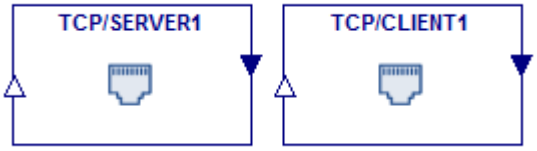


Рис. 76. Блоки **TCPServer** и **TCPClient**

TCPServer является своеобразным хранилищем данных, для обмена ими между клиентами. Клиенты, подключенные к серверу, получают доступ к этому хранилищу, и могут записывать свои данные, или же получать данные записанные в сервер другими клиентами. Данные на сервер хранятся в индексированных разделах. Индекс раздела называется адресом. Для чтения или записи данных в раздел сервера, в запросе клиента на сервер отправляется соответствующий разделу адрес.

TCPClient реализует доступ к серверу, для чтения и записи данных.

Пример создания простой связки двух сред **MexBIOS Development Studio**.

1. Запустите **MexBIOS Development Studio.**
2. Вынесите на поле набора компонент **TCP/SERVER.**
3. Вынесите на поле набора компонент **TCP/CLIENT.**
4. Раскройте формулу **MODELS_MAIN.** Обратите внимание: палитра изменилась.
5. Раскройте раздел **Embedded→Controls** палитры.
6. Вынесите на поле набора блок **PROTO_IN**.
7. В раскрывшемся контекстном меню выбрать **TCP/CLIENT1**.
8. Вынесите на поле набора блок **PROTO_OUT**.
9. В раскрывшемся контекстном меню выбрать **TCP/CLIENT1**.
10. Вынесите на поле набора блок **LABEL**.
11. Вынесите на поле набора блок **TRACKBAR**.
12. Соедините блоки **PROTO_IN** и **LABEL** между собой соединительной линией.
13. Аналогичным образом соедините между собой блоки **PROTO_OUT** и **TRACKBAR**.

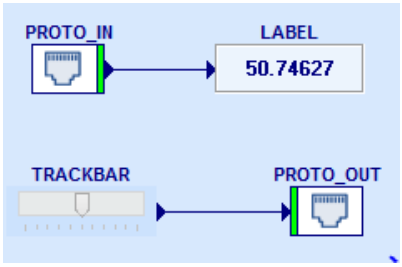


Рис. 77. Собранная схема для тестирования работы блоков протокола TCP

14. Запустите обновление.
15. В случае удачного соединения блоки **PROTO_IN** и **PROTO_OUT** подсветятся зеленым (как показано на рис. 77).
16. Остановите обновление.
17. Добавьте еще один блок **PROTO_OUT** и одни блок **TRACKBAR**.
18. Соедините новые блоки между собой.
19. Настройте новый блок **PROTO_OUT** для записи данных по адресу **0x1.**
20. Запустите еще один экземпляр **MexBIOS Development Studio**.
21. Вынесите на поле набора компонент **TCP/CLIENT**.
22. Раскройте формулу **MODELS_MAIN**.
23. Вынесите на поле набора блок **PROTO_IN**.
24. Настройте новый блок **PROTO_ IN** для чтения данных по адресу **0x1**.
25. Вынесите на поле набора блок **LABEL**.
26. Соедините блоки **PROTO_IN** и **LABEL** между собой соединительной линией.
27. Запустите обновление обоих сред **MexBIOS Development Studio**.
28. В случае удачного соединения данные **TRACKBAR** одной среды будут передаваться в **LABEL** другой.

В качестве TCP клиента для компонента **TCPServer** могут выступать и другие приложения, в которых реализован прикладной протокол связи среды **MexBIOS™ Development Studio**.

Для обеспечения возможности использования протокол связи среды **MexBIOS™ Development Studio** в своем приложении подключите к исходному коду своего приложения модуль **UTcpMessages**. В заголовочном файле модуля объявлена структура

```
struct TTcpPacket {
 ULONG func;   // функция
 ULONG addr;   //адрес
 ULONG data;   //данные
 ULONG status; //статус
 ULONG crc;    //контрольная сумма
};
```

Пример формирования структуры для чтения данных по адресу 0x1

```
TTcpPacket sendPacket;
 sendPacket.func  = TCP_FUNC_READ;
 sendPacket.addr  = 0x1;
 sendPacket.data  =0;
 sendPacket.status =0;
 sendPacket.crc  = calcCRC32((LPSTR)&sendPacket, TCP_PACKET_DATA_SIZE);
send(socket, (LPCSTR)&sendPacket, sizeof(TTcpPacket), 0);
```

Пример формирования структуры для записи данных по адресу 0x1

```
TTcpPacket sendPacket;
 sendPacket.func  = TCP_FUNC_WRITE;
 sendPacket.addr  = 0x1;
 sendPacket.data  =0;
 sendPacket.status =0;
 sendPacket.crc  = calcCRC32((LPSTR)&sendPacket, TCP_PACKET_DATA_SIZE);
 recv(socket, (LPSTR)packet, sizeof(TTcpPacket), 0);
```

Пример простого TCP клиента

```
#include <stdio.h>
 #include <string.h>
 #include <winsock2.h>
 #include <windows.h>

 #define PORT 5001
 #define SERVERADDR "127.0.0.1"

 int main(int argc, char* argv[])
 {
char buff[1024];
  if (WSAStartup(0x202,(WSADATA *)&buff[0]))
  {
   ShowMessage("WSAStart error %d\n"+IntToStr(WSAGetLastError()));
   return ;
  }

  SOCKET my_sock;
  my_sock=socket(AF_INET,SOCK_STREAM,0);
  if (my_sock < 0)
  {
   ShowMessage("Socket() error %d\n"+IntToStr(WSAGetLastError()));
   return ;
  }

  sockaddr_in dest_addr;
  dest_addr.sin_family=AF_INET;
  dest_addr.sin_port=htons(PORT);
  HOSTENT *hst;

  if (inet_addr(SERVERADDR)!=INADDR_NONE)
   dest_addr.sin_addr.s_addr=inet_addr(SERVERADDR);
  else
   if (hst=gethostbyname(SERVERADDR))
   ((unsigned long *)&dest_addr.sin_addr)[0]=
    ((unsigned long **)hst->h_addr_list)[0][0];
   else
   {
    printf("Invalid address %s\n",SERVERADDR);
    closesocket(my_sock);
    WSACleanup();
    return ;
   }

  if (connect(my_sock,(sockaddr *)&dest_addr,
        sizeof(dest_addr)))
  {
   ShowMessage("Connect error %d\n"+IntToStr(WSAGetLastError()) );
   return;
  }

 TTcpPacket sendPacket;
 sendPacket.func  = TCP_FUNC_READ;
 sendPacket.addr  = 0x0;
 sendPacket.data  = 100;
 sendPacket.status = 0;
 sendPacket.crc  = calcCRC32((LPSTR)&sendPacket, TCP_PACKET_DATA_SIZE);
 send(my_sock, (LPCSTR)&sendPacket, sizeof(TTcpPacket), 0);
        recv(my_sock, (LPSTR)&sendPacket, sizeof(TTcpPacket), 0);
  ShowMessage(IntToStr(sendPacket.data));
       closesocket(my_sock);
WSACleanup(); return ; }
```

Моделирование/Работа с чипом

В **MexBIOS™ Development Studio** предусмотрены два режима работы программы: моделирование и работа с чипом.

Для перехода в режим моделирования необходимо нажать кнопку **Начать моделирование**. После чего кнопка запуска превратится в кнопку останова моделирования **Остановить моделирование**.

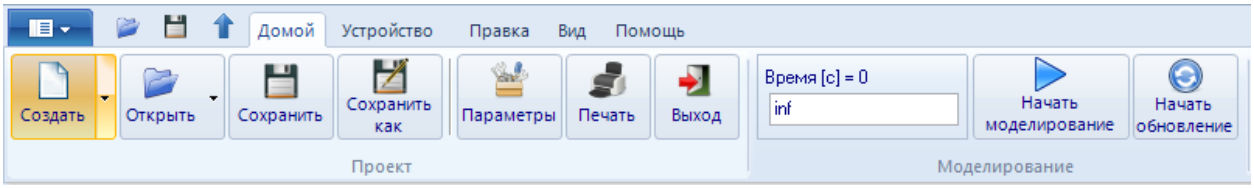


Рис. 78. Панель управления активными режимами

Моделирование (симуляция)

Моделирование – это фактически процесс моделирования на компьютере работы программного обеспечения в контроллере с **MexBIOS™** в отсутствии реального контроллера.

Этот активный режим характеризуется выполнением программы в адресном пространстве среды **MexBIOS™ Development Studio** на встроенной модели ядра **MexBIOS™**. При этом **MexBIOS™ Development Studio** имеет доступ к массиву глобальных данных.

Работа с чипом

Работа с чипом – это процесс управления и наблюдения за работой реального устройства с загруженной программой **MexBIOS™**, управляющего реальным процессом. Этот режим характеризуется выполнением программы в управляющем контроллере.

Для перехода в этот режим необходимо **установить связь** с микроконтроллером и запустить обновление (кнопка **Начать обновление**).

В этом режиме **MexBIOS™ Development Studio** не имеет доступа к массиву глобальных данных **MexBIOS™** устройства напрямую. Обновление окна **Переменные**, осциллографа и визуальных органов происходит с использованием транспортного протокола через регулярные запросы к устройству, при этом пользователь получает возможность влиять на работу программы в контроллере под управлением **MexBIOS™** в реальном времени, путём изменения переменных программы в окне **Переменные**.

Модели физических объектов в режиме работы с чипом

Для моделирования физических объектов в среде **MexBIOS™ Development Studio** существует библиотека **Models**. Модели предназначены для использования вне **MexBIOS™**, то есть они не являются частью программы. Назначение моделей физических объектов – имитация работы реальных физических устройств в режиме моделирования и отладка создаваемой программы, перед запуском созданной программы на контроллере совместно с реальным объектом управления.

Связь между библиотекой **Models** и библиотекой контроллера осуществляется блоками **TP_IN** (приём данных), **TP_OUT** (передача данных).

Принципы создания программ

Структура и условие выполнения частей программы создаются с помощью внутренних блоков (**EVENT**, **FORMULA**, **ALGORITHM**, **WHILE**, **STATE_FLOW**). Ветвление структуры (выполнение одной из частей программы по условию) программы производится с помощью блока условия IF. Внутренние блоки соединяются с помощью линий связи, которые показывают направление и порядок выполнения соединённых внутренних блоков.

Основные функции программы создаются в блоках **FORMULA** и **STATE** из внешних блоков библиотеки для выбранного процессора с помощью настройки параметров и создания связей между блоками. Связь между функциями программы (передача данных) осуществляется с помощью блоков **TP_IN**, **TP_OUT**. Также из разных **FORMULA** и **STATE** можно обратиться к одной переменной **VAR**.

После создания нового файла по умолчанию создаётся **FORMULA** (**MAIN**), присоединённая в ветку **BACKGROUND**. Все элементы программы, подключенные к ветке **BACKGROUND**, будут находиться в цикле не занятости процессора (**Idle Loop** – т.е. будут выполняться в свободное от выполнения прерываний время процессора). Вид ветки представлен на следующем рисунке:

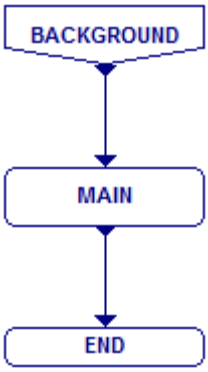


Рис. 79. Начальная структура программы

В ветке **BACKGROUND** блоки, требующие работы со строгой периодичностью, будут работать без привязки к реальному времени, т.е. результаты могут быть не корректны.

Цикл занятости процессора формируется прерываниями (Interrupts). Прерывание создаётся с помощью блока **EVENT** с соответствующей настройкой (см. далее).

Линия связи **EVENT** с **FORMULA**, **ALGORITHM**, **WHILE**, **STATE_FLOW**, **IF** передаёт очерёдность выполнения блоков с предыдущего элемента программы, образуя тем самым структуру программы. Также необходимо расставлять очерёдность выполнения внешних блоков. Для этого необходимо в порядке нужной очерёдности выполнения блоков выделить блоки, находящиеся на схеме. Далее нажать ПКМ по выделенным блокам и выбрать пункт меню **Задать порядок вызова**. Для отображения/скрытия индексов очерёдности выполнения блоков нажмите клавишу **F3**.

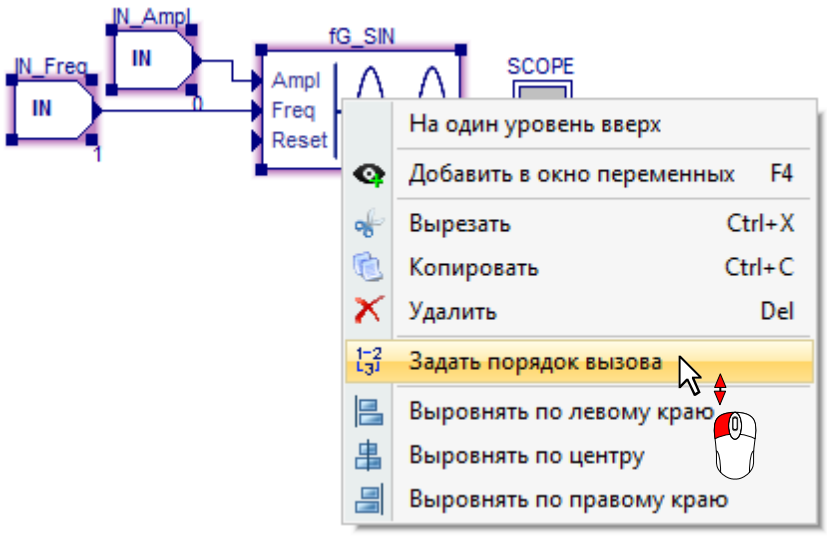


Рис. 80. Назначение порядка выполнения блоков

Процессором выполняются только те конструкции, которые подключены к **EVENT** (при срабатывании условия запуска события). Не подключенные внешние блоки к **EVENT** не генерируются в матрицу и, следовательно, не выполняются.

К **EVENT** и каждой последующей системе может быть подключена только одна система. Блок **IF** разделяет алгоритм на два возможных пути, к выходу **true** и **false** можно также последовательно подключать блоки **FORMULA**, **ALGORITHM**, **STATE FLOW**.

Создание программы

Любой из внутренних блоков программы **ALGORITHM**, **FORMULA**, **WHILE**, **STATE_FLOW** должен быть непосредственно, либо через другие внутренние блоки, присоединён к блоку **EVENT**. При создании схемы на поле набора помещена структура (**BACKGROUND**), которая работает в цикле незанятости процессора (**Idle loop**). В цикле незанятости могут располагаться блоки, не требующие строгой периодичности выполнения. Блоки, которые требуют строгой периодичности выполнения, должны быть соединены с блоком **EVENT**, который привязан к периодичному прерыванию.

Рассмотрим порядок действий для создания периодически выполняемой программы:

1. Для создания ветки программы, привязанной к прерыванию, вынесите на поле набора блок **EVENT**.

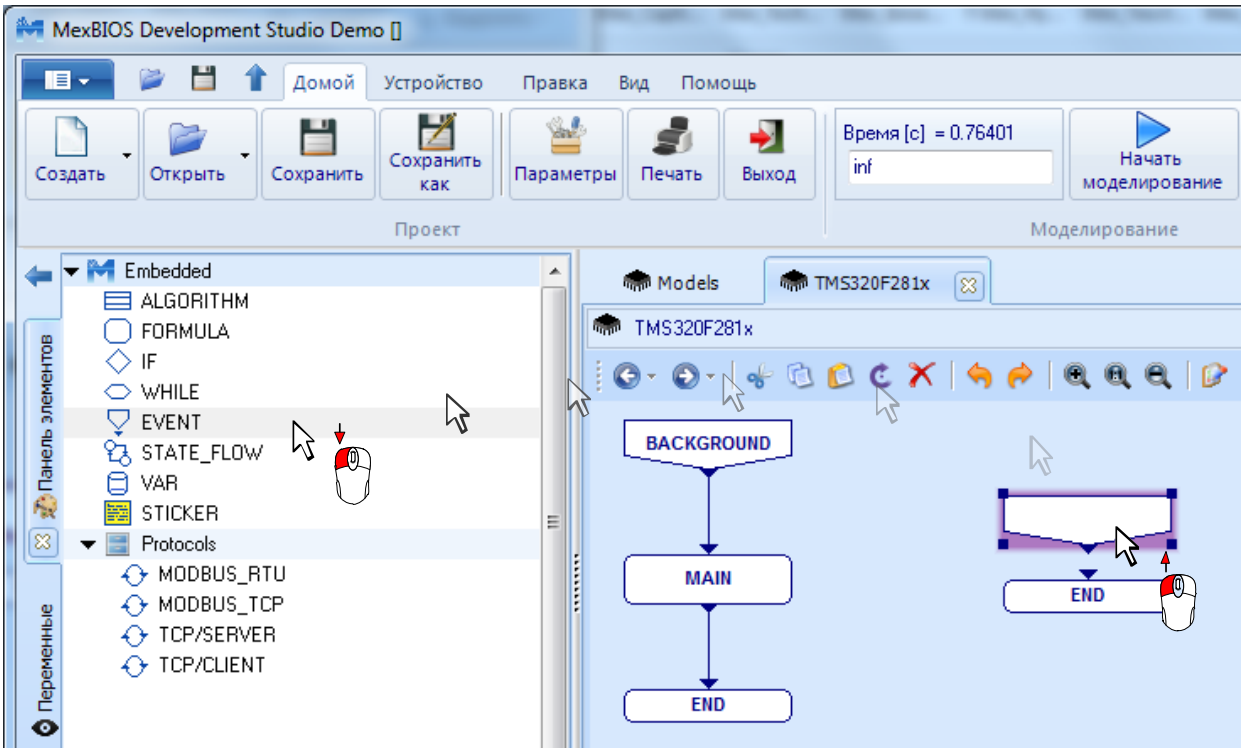


Рис. 81. Вынесение блока **EVENT** на поле набора

2. Для настройки свойств блока **EVENT** в инспекторе выделите блок и проделайте следующее:
   2.1. Свойство **Источник** переключите в **2 : Аппаратное прерывание.**
   2.2. Выберете вектор прерывания **TINT0** (для разных процессоров может быть разным, уточнить в документации к библиотеке процессора), раскрыв список **Вектор**.
   2.3. Установите период прерывания, равный периоду прерывания указанному в стартовом проекте. Для процессора **TMS320F281x** частота прерывания TINT0 равна 5000 Гц, задаётся в файле **config.h** стартового проекта:

```
#define HZ                              5000  // Main isr frequency
```

   Период = 1/5000 = 0.0002 – за одну секунду программа привязанная к главному прерыванию выполнится 5000 раз.

3. Свойство **Режим моделирования** установить в режим **2: Непрерывное**. Настройка **0: Выключено** означает, что событие не будет запускаться, **1: Однократное** - выполнится один раз.

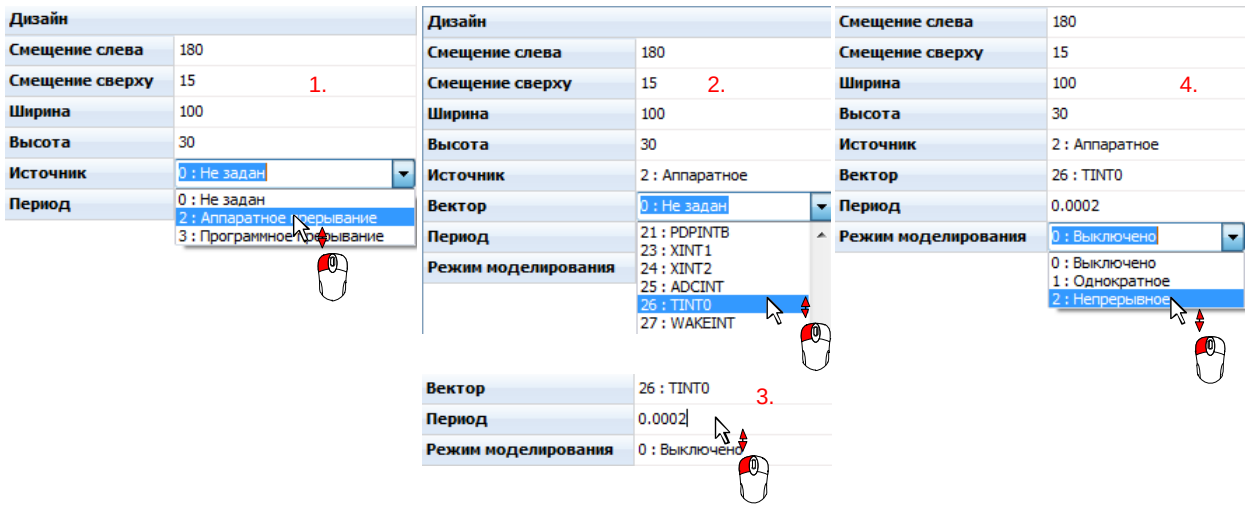


Рис. 82. Настройка блока **EVENT**

4. Полученная схема имеет вид:

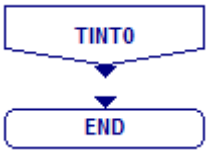


Рис. 83. Настроенное на прерывание событие

Прерывание настроено. В зависимости от необходимости к событию **EVENT**, привязанному к прерыванию **TINT0** можно последовательно присоединить **FORMULA**, **ALGORITHM**, **STATEFLOW**, **IF**, **WHILE**. Для проверки работоспособности проделаем следующее:

5. Перетащите на поле набора аналогичным образом, из **Панели элементов**, группа **Embedded**, блок **FORMULA**.

6. Присоединить вход к выходу блока **TINT0**, выход присоединить к блоку **END**. Полученная схема примет вид:

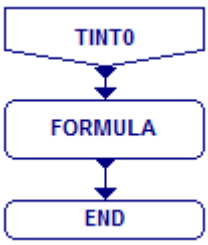


Рис. 84. Подключение к EVENT блока FORMULA

7. Двойным нажатием **ЛКМ** войти внутрь **FORMULA**.

8. Собрать следующую схему с использованием **Label (Embedded→Controls→LABEL)**.

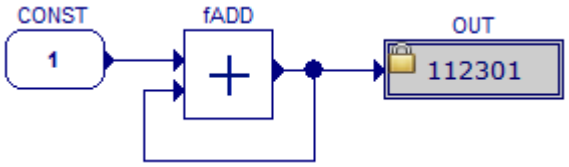


Рис. 85. Схема счётчика

8.1. Блок **CONST** находится в **Панели инструментов→Embedded** (активно поле набор FORMULA)**.** Параметры блока CONST: **Формат 0: Integer**, **Значение = 1**;

8.2. Блок **LABEL** находится в **Панели инструментов →Embedded→Controls.** Параметры блока **LABEL**: **Формат 0: Integer**, **Переменная 0: Не задано**, **Значение = 0**;

8.3. Блок **ADD** находится в **Панели инструментов →TMS320F281x→Math.**

9. Запустить моделирование программы (горячая клавиша **F5** либо вкладка **Домой→Начать моделирование)**. Убедиться, что счётчик аккумулирует единицу при каждом такте расчёта. За единицу модельного времени счётчик должен досчитать до 5000.

10. Если значение в **Label** при моделировании не изменяется, то проверьте что в **CONST** параметр **Значение = 1** и в настройках EVENT TINT0 параметр **Режим моделирования** установлен как **2 : Непрерывное**.

Глобальные переменные VAR

Объявление переменной (вынос блока VAR на рабочее поле) выделяет ячейку памяти. К выделенной ячейке памяти можно обратиться из любого места программы (в рамках одной библиотеки), то есть произвести чтение и запись в переменную. Переменные VAR могут применяться для управления блоками IF, WHILE, создания программного прерывания блоком EVENT, организацию перехода из одного состояния в другое в STATE_FLOW. Также могут применятся для передачи сигналов в программе по усмотрению пользователя.

Блок VAR работает совместно с блоками IN и OUT. Привязка этих блоков к глобальной переменной осуществляется выбором имени переменной из списка существующих в параметре **Переменная**, как показано на рис. 86.

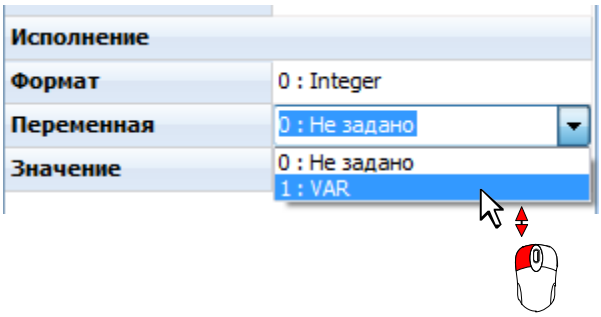


Рис. 86. Привязка блоков IN и OUT к переменной VAR

Если привязать блок OUT к переменной VAR, то значение, записываемое в блок OUT, будет передаваться и в VAR (рис. 87). Изменить значение блока VAR в этом случае можно только изменением величины на входе блока OUT.

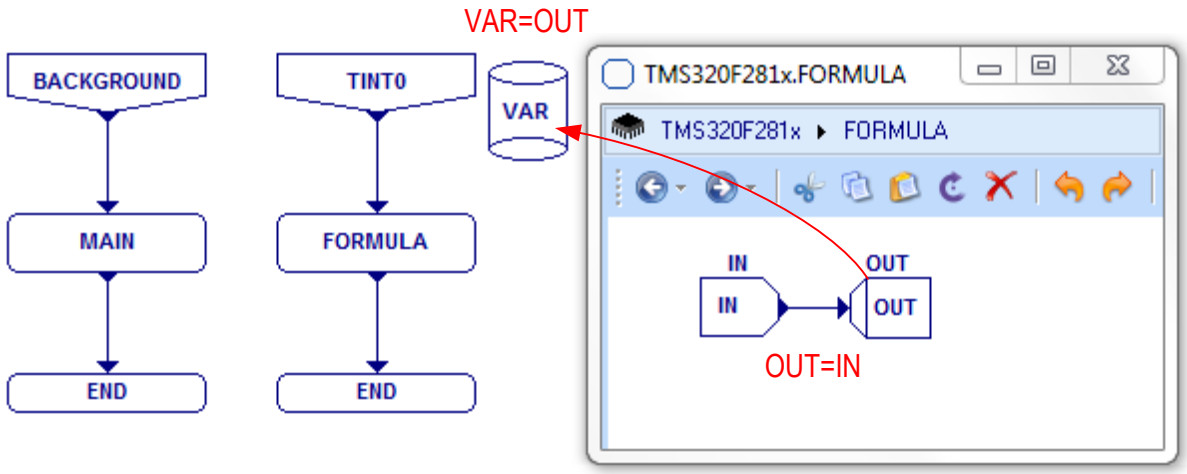


Рис. 87

Внешний вид, привязанных к переменным, блоков **IN** и **OUT** изменяется.

Если привязать блок **IN** к переменной **VAR**, то на схему с выхода блока **IN** будет передаваться значение переменной **VAR**.

Задать значение **VAR** или **IN** (связанные между собой) вручную через вкладку окна

Переменные можно только при отсутствии привязки переменной **VAR** к блоку **OUT**.

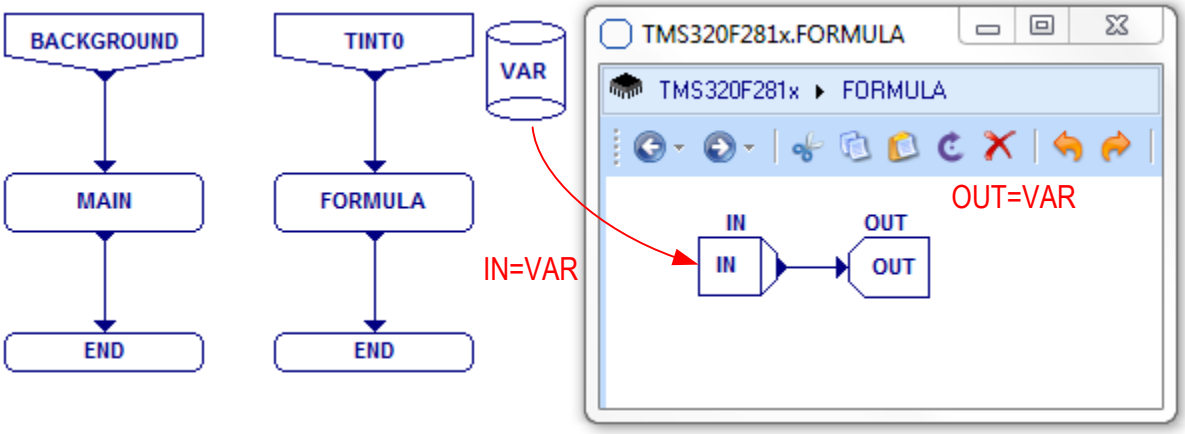


Рис. 88. Переменая и связанный с ней блок **IN**

Применение блока IF

Расчёт части программы на частоте кратной основной частоте прерывания

1. Для деления частоты нужно использовать специальный импульсный генератор FREQ_DEVIDER (**Панель элементов** группа **Sources**). Для этого в FORMULA, которая выполняется с частотой главного прерывания, нужно поместить схему:

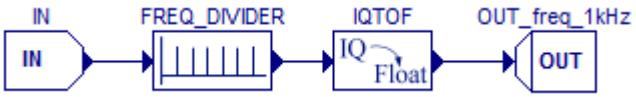


Рис. 89. Использование FREQ_DEVIDER

2. В блоке IN указать число, кратно которому необходимо чтобы выполнялась часть программы. Выход необходимо передать в OUT, привязанной к переменной VAR.
3. Собрать схему простого счётчика

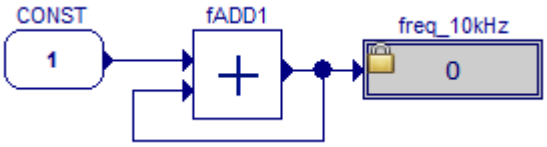


Рис. 90. Простой счётчик

3.1. Блок **CONST** и **LABEL** находятся в группе **Embedded**. **fADD** – группа Math.
3.2. Задать время симуляции 1 с. Запустить симуляцию – убедиться, что за 1 с счётчик накапливает значение 10000 (частота TINT0 для данного примера).

4. Далее собрать следующую схему:

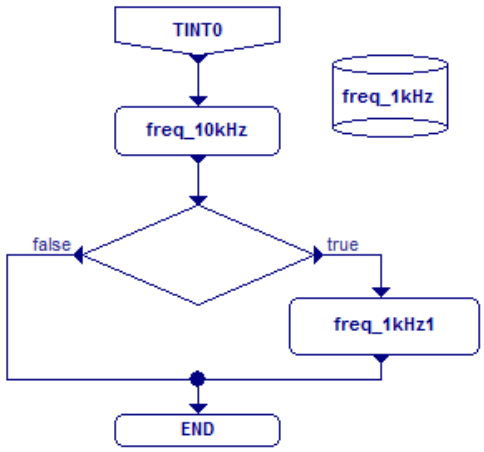


Рис. 91. Схема делителя частоты прерывания

В блоке freq_10kHz располагается схема делителя частоты. А в блоке freq_1kHz будет выполняться на кратной основной частоте.

4.1. Блоки **FORMULA**, **IF**, **EVENT**, **VAR** находятся в **Панель элементов→Embedded**.

4.2. Блок **IF** необходимо настроить следующим образом:

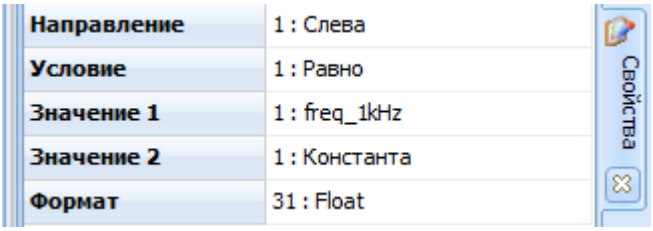


Рис. 92. Настройка блока IF

4.3. Параметр **Условие** задаёт условие **Равно**, при котором срабатывает ветка **true**. Величину **Значение 1** привязать к переменной, а **Значение 2** задать константой и равной 1. Формат оставить Integer.

Внимание: Формат переменных должен быть одинаковым и должен совпадать с форматом, заданным в параметре **Формат**.

4.4. После назначения условия внешний вид блока **IF** отображает назначенное условие.

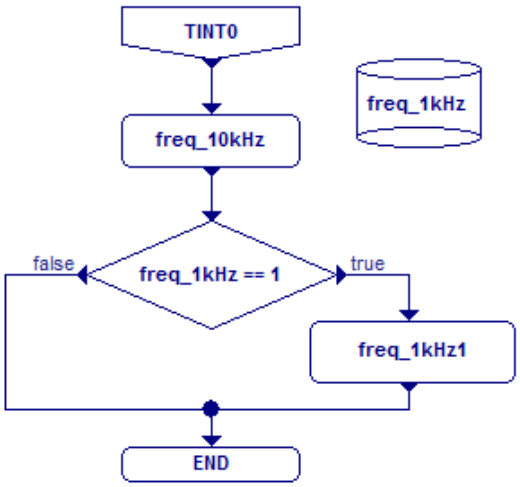


Рис. 93. Собранная схема

4.5. Скопируйте из freq_10kHz схему простого счётчика и вставьте в freq_1kHz.
4.6. Добавьте LABEL из freq_10kHz и freq_1kHz в окно Переменные.
4.7. Запустите симуляцию. Убедитесь, что накопленное число в счётчике freq_1kHz меньше в N раз чем в freq_10kHz.

Применение State Flow

Блок **STATE_FLOW** предназначен для реализации визуальной программы методом состояний.

Метод состояний (**state flow**) – метод программирования, когда задача описывается конечным числом состояний, в которых может работать программа для решения конкретной задачи. Переход из одного состояния в другое происходит по заданному событию перехода.

Блок может быть добавлен на главное поле набора, или в **ALGORITHM**. Внутри блока могут находиться блоки **STATE** и **STICKER**.

Пусть необходимо создать программу, выполняющую следующие действия:

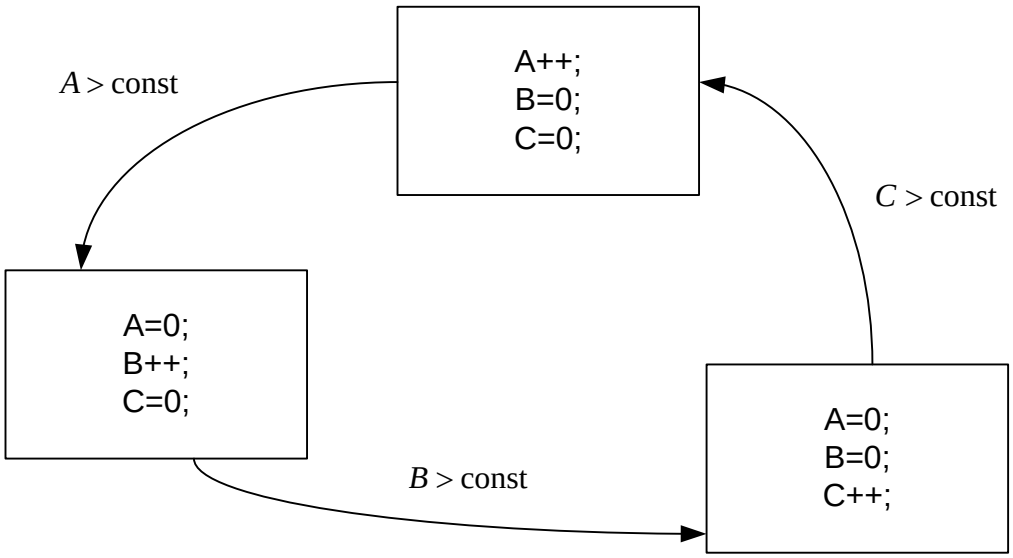


Рис. 94. Блок схема алгоритма

Для реализации необходимо три переменные: **A**, **B**, **C**. И ещё одна переменная **Const**.

Необходимо выполнить следующие действия:

1. Запустите программу и создайте схему для нужного процессора, например **TMS320F2833x**.
2. Из **Панель элементов→Embedded** добавьте четыре блока **VAR**. Переименуйте их согласно рис. 94 . В переменную **Const** параметр **Значение=100**;
3. Создайте прерывание **TINT0**, как описано в разделе «**Создание программы**».
4. Из **Панель элементов →Embedded** добавьте блок **STATE_FLOW**.
5. Полученная промежуточная схема представлена на рис. 95.

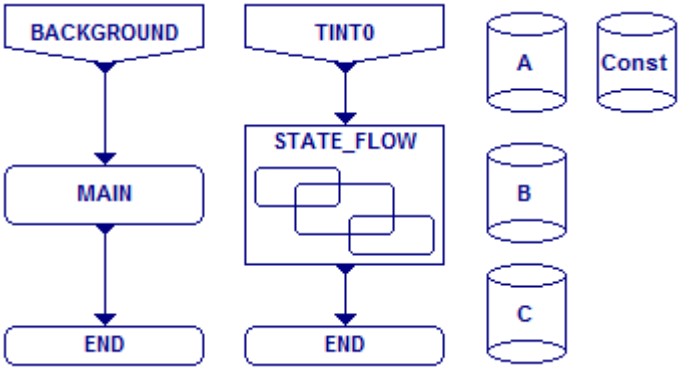


Рис. 95. Вид собранной схемы

6. Два раза нажмите ЛКМ на блок STATE_FLOW. Откроется поле набора.
   6.1. Добавьте три состояния из **Панель элементов** →**Embedded**→**STATE**. Первое вынесенное на поле набора состояние установится начальным состоянием в параметре **Начальное состояние** блока **STATE_FLOW**.
   6.2. Соедините блоки **STATE** между собой.
   6.3. Два раза нажмите ЛКМ на блоке **STATE** (начальное состояние).
   6.4. Внутри блока соберите следующую схему:

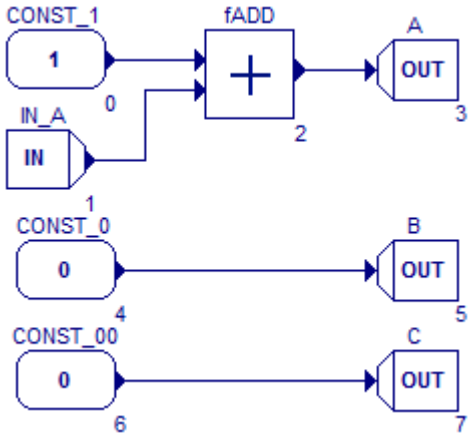


Рис. 96. Схема обработки состояния после назначения связи с переменными

   6.5. Для блоков на рис.97 произведите следующие настройки:

Блоки констант:
CONST_1: **Формат** 31: Float; **Значение** =1;
CONST_0: **Формат** 31: Float; **Значение** =0;
CONST_00: **Формат** 31: Float; **Значение** =0.
Блок IN:
IN_A: **Формат** 31:Float; **Значение**=1.
Блоки OUT:
A: **Формат** 31: Float; **Variable**: A; **Значение**=0;
B: **Формат** 31: Float; **Variable**: B; **Значение**=0;
C: **Формат** 31: Float; **Variable**: C; **Значение**=0.

После назначения связи с переменной вид блоков OUT изменится.

В схеме, на рис. 96, счётчик реализует операцию инкрементирования A++.

6.6. Необходимо, чтобы порядок выполнения блоков был по направлению распространения сигналов от IN к OUT.

6.7. В другие состояния, необходимо скопировать полученную схему и поменять местами переменную, в которую будет производиться инкрементирование. Изменить переменную на счётчик. Схемы для двух других состояний представлены на рис. 97:

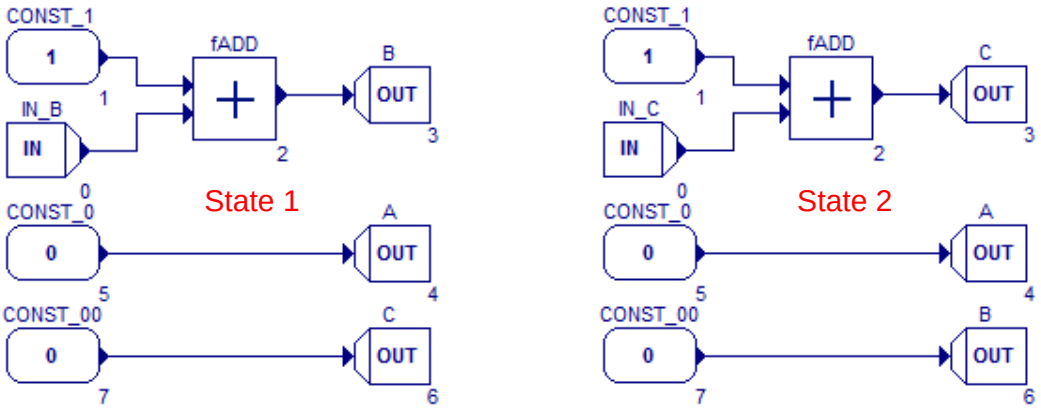


Рис. 97. Схемы для двух других состояний

Настройки для блоков констант не изменяются. Для второго состояния необходимо к счётчику подключить **OUT**, привязанный к блоку **B**, к входу счётчика подключить блок **IN**, привязанный к переменной **B**. Для третьего состояния необходимо к счётчику подключить **OUT**, привязанный к блоку **C**, к входу счётчика подключить блок **IN**, привязанный к переменной **C**.

7. Далее необходимо назначить условие на линии перехода из состояния в состояние.

7.1. Выделить нужную линию перехода внутри блока STATE_FLOW.

7.2. В **Свойства** назначить условие перехода, как показано на рис. 98:

Дизайн	
Условие	4 : Больше или равно
Значение 1	1 : A
Значение 2	4 : Const
Формат	31 : Float
Комментарий	

Рис. 98. Настройка условий перехода между состояниями

7.3. После назначения всех условий схема примет вид:

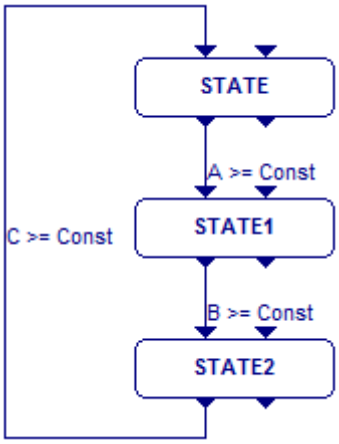


Рис. 99. Итоговая схема

8. Добавить переменные A, B, C в окне **Переменные**.
9. В вкладке **Домой→Параметры.** В окне **Параметры** на вкладке **Общие** в разделе **Моделирование** установить настройки

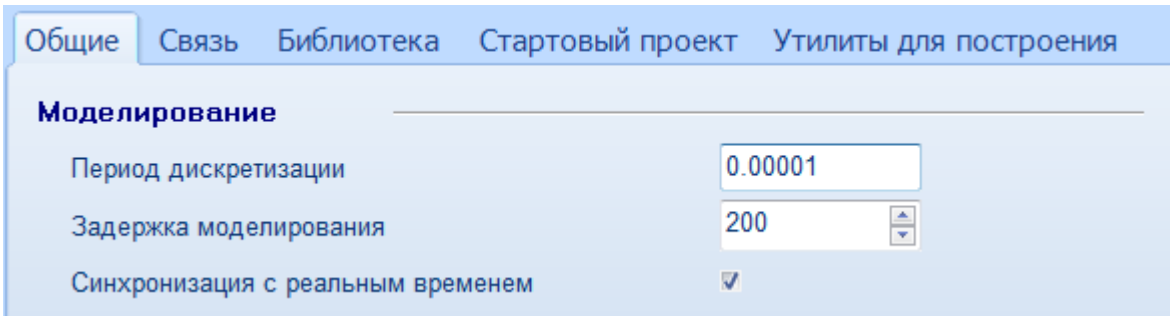


Рис. 100. Настройки проекта для иллюстрации работы STATE_FLOW

10. Запустить моделирование системы. Наблюдать в окне **Переменные** работу алгоритма.

Применение блока цикла WHILE

Цикл (WHILE) – отличается от встроенного блока **ALGORITHM** тем, что выход из системы созданной внутри этого блока может произойти только по заданному в параметрах блока условию.

Важно, чтобы условие выхода из блока **WHILE** срабатывало за ограниченное число циклов, иначе произойдёт зависание программы в режиме симуляции и зависание ядра в микроконтроллере.

Одной из особенностей данного цикла является то, что во время выполнения цикла невозможно в реальном времени отслеживать процессы, происходящие внутри него. Обновление переменных, задействованных в цикле, производится после выхода из него.

Рассмотрим работу данного блока на примере.

Пример использования блока While.

Пример будет реализовывать следующее: в блоке **WHILE** будет производиться инкрементирование переменной **VAR**, пока не выполнится условие. После выхода из **WHILE** величина **VAR** будет накапливаться в **OUT** (с помощью сумматора и переменной). Переменная **VAR** в конце цикла будет обнуляться. Параллельно будет работать счётчик, показывающий работу основного прерывания. Например, если условие выхода из **WHILE** будет **VAR<10**, то схемы внутри **WHILE** будут рассчитываться **10** раз за один такт события (прерывания), к которому подключен блок **WHILE**.

1. Запустите программу и создайте схему для нужного процессора, например **TMS320F281x**.
2. Создайте прерывание **TINT0**, как описано в разделе «**Создание программы**».
3. Из **Панель элементов→Embedded** добавьте блок **WHILE**.
4. Добавьте на поле набора блок **FORMULA** из библиотеки **Embedded**.
5. Добавьте на поле блок **VAR**.
6. Задайте свойства блока **WHILE** в соответствии с Рис. 101:

Имя	WHILE
Смещение слева	220
Смещение сверху	70
Ширина	100
Высота	30
Условие	5 : Меньше
Значение 1	1 : VAR
Значение 2	10 : Константа
Формат	0 : Integer

Рис. 101

7. Соедините и переименуйте элементы схемы в соответствии с рис. 102:

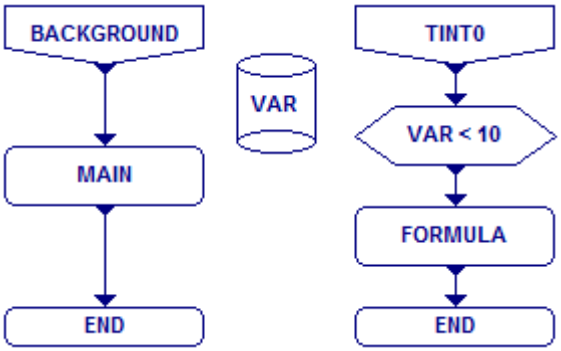


Рис. 102

8. Нажмите два раза ЛКМ по блоку **WHILE,** откроется поле набора цикла While.
9. Добавьте блок **FORMULA** из **Palette**, произведите соединение как показано на рис. 103:

Рис. 103

10. Перейдите на поле набора **FORMULA_WHILE**.
11. Создайте внутри блока **FORMULA_WHILE** схему, аналогичную представленной на рис. 104:

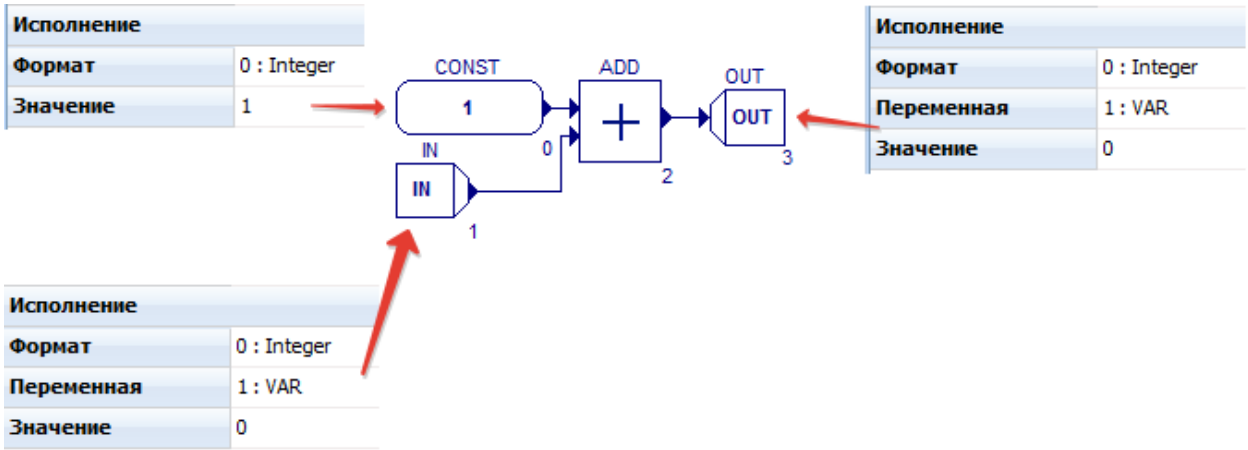


Рис. 104. Настройки в схеме внутри **WHILE / FORMULA**

12. Перейдите в окно набора формул блока **FORMULA**, находящегося в корневом поле набора и создайте в нем схему, аналогичную приведённой на рис. 105:

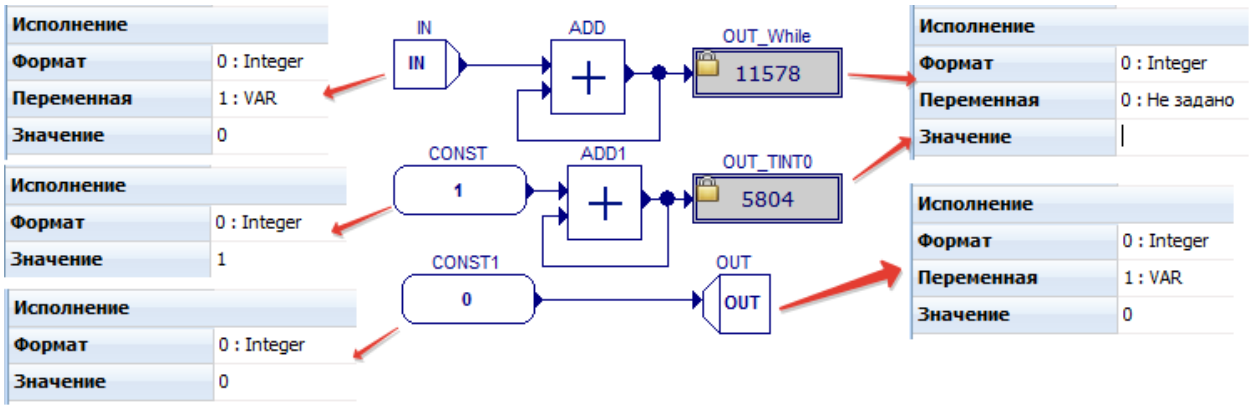


Рис. 105. Схема счётчиков

13. Задайте параметр **Замедление моделирования** равным 100.

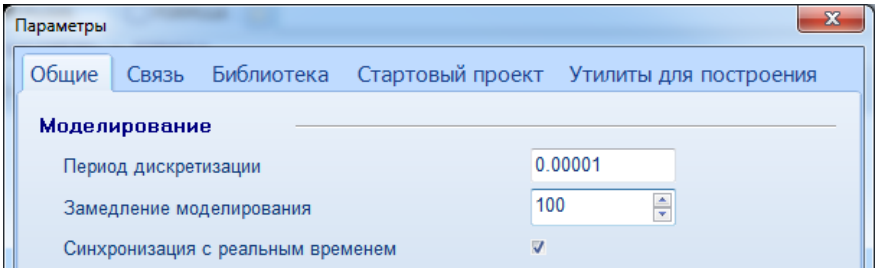


Рис. 106. Настройка окна **Параметры**

14. Порядок выполнения блоков, должен быть по порядку в направлении передачи сигнала (от **IN** к **OUT**).
15. Запустите симуляцию и наблюдайте за изменением отображаемых значений блоков **LABEL**:

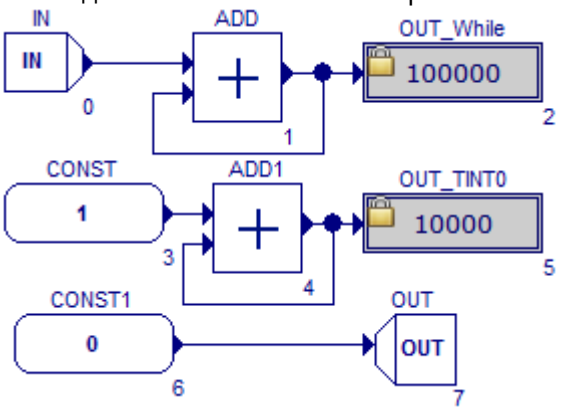


Рис. 107. Схема работы **WHILE**

В **WHILE** происходит увеличение переменной **VAR** на единицу за один такт выполнения **WHILE** до тех пор, пока выполняется условие цикла (**VAR<10**). По достижении переменной **VAR** значения **10** выполняется условие выхода из цикла и происходит переход в блок **FORMULA**_1, где к счетчику прибавляется значение переменной **VAR**. Второй счётчик показывает количество выполненных прерываний. После достижения счётчиком заданного значения в условии WHILE, переменная **VAR** обнуляется. Цикл программы завершается.

Визуальные компоненты управления

Визуальные органы управления являются надстройками над блоками **IN** и **OUT**. К каждому блоку **IN**, **OUT** можно назначить визуальный орган управления. Визуальные органы управления позволяют сделать программное обеспечение **MexBIOS™ Development Studio** интерактивным.

В **MexBIOS™ Development Studio** существуют два набора визуальных органов управления, для ввода и вывода информации, соответственно назначаемые блокам **IN** и **OUT**.

IN

- None;
- BUTTON;
- TRACKBAR;
- EDIT;
- RADIO_IN;
- CHECK_IN;
- PROTO_IN (появляется при наличии в проекте одного или нескольких блоков протокола).
- GRID;
- STRING_IN;

Out

- None;
- LABEL;
- PROGRESS;
- BULB;
- AGAUGE;
- ANIMATE;
- RADIO_OUT;
- CHECK_OUT;
- PROTO_OUT (появляется при наличии в проекте одного или нескольких блоков протокола).
- STRING_OUT

Назначение визуальных органов управления для других типов блоков не предусмотрено.

Когда схема находится в неактивном режиме, на визуальных органах управления появляется значок «замок», показывающий, что визуальный орган управления не активен.

В неактивном режиме можно произвести изменение состояния визуального органа управления. Для этого необходимо зажать кнопку Alt и нажать ЛКМ один раз по блоку – блок перейдёт в активное состояние, кнопку Alt можно отпустить. Провести изменение состояния визуального органа управления с помощью соответствующего нажатия ЛКМ. После нажать на поле набора ЛКМ, визуальный орган управления перейдёт в неактивное состояние.

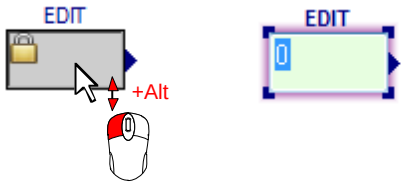


Рис. 108. Активация визуального органа управления для редактирования

1. **Назначение визуального органа управления.**
   1.1. Вызвать контекстное меню правым кликом мыши по блоку.
   1.2. В разделе **Добавить компонент** выбрать нужный орган управления.

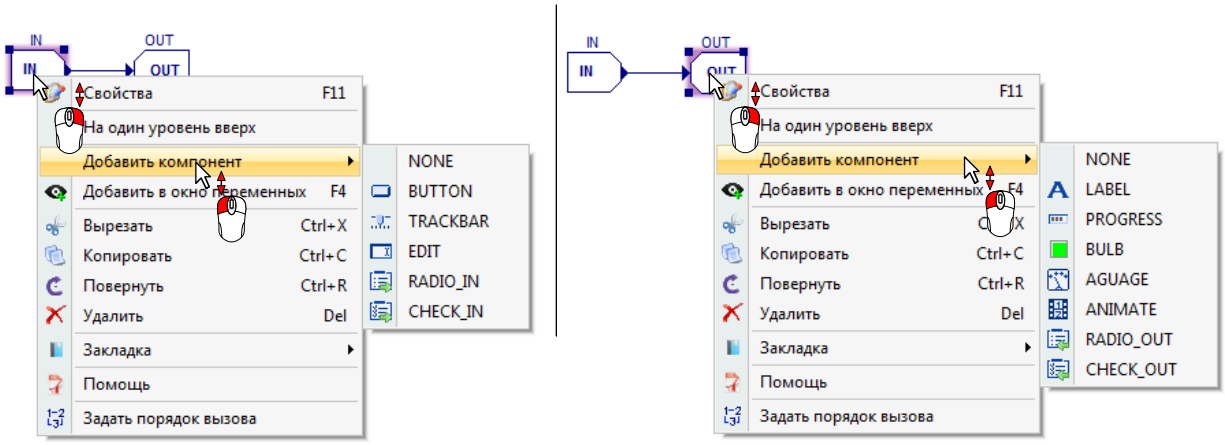
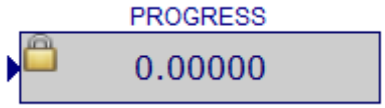
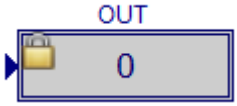
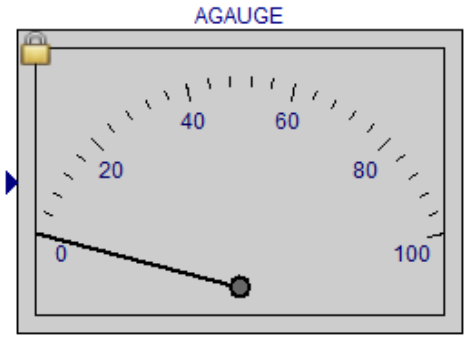
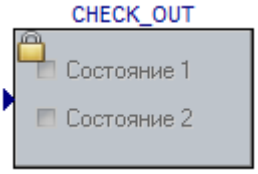
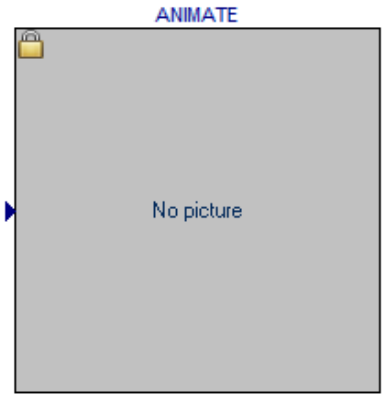


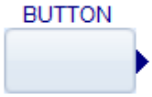
Рис. 109. Назначение визуального органа управления для блока IN и OUT

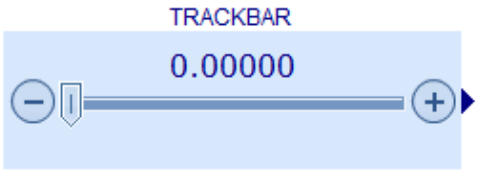
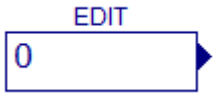
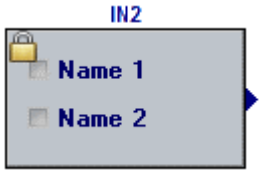
Тип органа	Свойства инспектора
None Отменяет назначенные ранее органы управления	
Визуальные компоненты для блока OUT	
Bulb  Служит для отображения дискретного сигнала. Два цвета соответствуют высокому и низкому дискретному сигналу соответственно	**Цвет 1:** Цвет логической 1 **Цвет 2:** Цвет логического 0 Возможно назначить изображения: **Изображение 1**: Логическая 1 **Изображение 2**: Логический 0 **Точность** **Период анимации** -

Progress bar  Служит для отображения уровня сигнала в процентном отношении в диапазоне минимума – максимума	**Минимум:** Значение соответствующее 0% заполнению; **Максимум:** Значение соответствующее 100% заполнению; **Точность** – число знаков после запятой для отображения; **Уровень 1** – число после превышения, которого цвет изменится на желтый; **Уровень 2** – число после превышения, которого цвет изменится на оранжевый; **Неопределённый** – полоса прогресса будет по циклу показывать изменение входного сигнала.
Label  Служит для вывода на поле набора текущего значения уровня сигнала в числовом формате сигнала.	Основа – назначение типа отображаемого числа. Возможно представить число в виде: **hex** – шестнадцатеричного числа; **dec** – десятичного числа; **bin** – двоичного числа; **oct** – восьмеричного числа; **char** – кода Char; **float** – числа с плавающей запятой; **scientific** – числа в формате scientific; **unsigned** – без знакового числа; **auto** – автоматическое назначение Основы; **qvalue** – число в формате IQ; **time** – время; **date** – дата; **Точность** – число знаков после запятой; Необходимо задать **Формат** числа.

Тип органа	Свойства инспектора
AGAUGE  Служит для отображения сигнала в стиле аналогового прибора	**Угол:** Угол дуги хода указателя между минимальным и максимальным значением, в градусах. **Минимум:** Минимальное значение аналогового прибора; **Максимум:** Максимальное значение аналогового прибора; **Количество делений:** количество меток между соседними значениями на шкале прибора **Точность:** точность прибора
Radio Group **радио группа OUT**  	Создаёт на поле радио группу. В свойстве **Элементы** необходимо написать нужное число радио кнопок в формате **N□:□ИмяЭлемента**. Где N – целое число, которое будет отображаться на выходе блока, при выборе пункта **ИмяЭлемента**, а □ – знак пробела, В зависимости от величины поступающего сигнала будет выделяться соответствующая кнопка радио группы.
Check Group **группа флагов OUT**  	Создаёт на поле группу флагов. В свойстве **Элементы** необходимо написать нужное число радио кнопок в формате **N□:□ИмяЭлемента** где N – целое число, а □ – знак пробела. В зависимости от величины поступающего сигнала будут выделяться соответствующие флаги.

Тип органа	Свойства инспектора
Animate 	Блок предназначен для отображения последовательности статических картинок. В зависимости от значения входного сигнала отображается та или иная картинка. **Папка с изображениями:** свойство для задания последовательности картинок в компонент. Картинки, предназначенные для загрузки в компонент, помещаются в одну директорию под своими именами. Имя файла картинки формируется следующим образом: [целое число].bmp. Число, указанное в имени файла картинки, показывает при каком значении сигнала на входе компонента, на компоненте отобразится именно эта картинка. Для загрузки в компонент доступны только файлы битовых матриц (.bmp). Для загрузки всех картинок из директории в компонент в открывшемся диалоге, при редактировании свойства Image, выбрать любую картинку доступную для загрузки. Вместе с ней в компонент загрузятся и все остальные, доступные для загрузки картинки из директории. **В папке Video lessons см. фильм ANIMATE.avi**
PROTO_OUT блок записи данных по адресу определённого протокола	См. раздел «Блоки организации коммуникации»
STRING_OUT блок для вывода данных типа string (текстовые данные)	**Длина строки** – число символов для отображения **Смещение –** начало отображение символов

Тип органа	Свойства инспектора
None Отменяет назначенные ранее органы управления	
Визуальные компоненты для блока IN	
Button  Кнопка. Служит для задания двух уровней сигналов, соответствующих нажатому и отпущенному состоянию	**Группа:** Индекс группы, к которой принадлежит кнопка. **Группа** = 0 – кнопка не принадлежит ни одной группе. Нажатие и возврат к исходному (не нажатому) состоянию за один клик мышью по кнопке. Кнопка хранит нажатое состояние, только пока нажата левая кнопка мыши. **Группа** ≠ 0 – кнопка переходит в режим «залипания». При клике мышью кнопка переходит в нажатое состояние. Возврат в исходное состояние происходит при повторном клике мышью. Отношение к одной группе означает, что кнопкам присвоено одинаковое число в параметре Группа. Внутри одной группы нахождение сразу нескольких кнопок в нажатом состоянии не допустимо. При переходе одной из кнопок в нажатое состояние все остальные автоматически переходят в не нажатое состояние. Нахождение всех кнопок в не нажатом состоянии внутри одной группы допустимо.  **Внимание**. Группа действует, даже если кнопки находятся в разных частях программы (FORMULA, STATE). **Значение «Отпущено»:** Значение, которое записывается в глобальный буфер при переходе кнопки в НЕ нажатое состояние. **Значение «Нажато»:** Значение, которое записывается в глобальный буфер при переходе кнопки в нажатое состояние. **Изображение:** изображении когда кнопка в положении «Отжато». **Изображение (Down):** изображение когда кнопка в положение «Нажато». **Изображение (Hot):** изображение когда курсор мыши наведён на кнопку.

TRACKBAR  Служит для задания уровня сигналов в заданном диапазоне.	**Значение:** Начальное положение движка **Минимум:** Минимальное значение положения движка **Максимум:** Максимальное значение положения движка **Шаг:** Шаг переключения движка Track Bar может управляться с клавиатуры, для этого необходимо выделить Track Bar в активном режиме. Стрелки вправо влево изменяют **Значение** на величину **Шаг**. **Точность** – число знаков после запятой, отображаемой Track Bar.
EDIT  Служит для прямого ввода значения параметра **Значение** на поле набора	**Минимум:** Ограничение на минимальное вводимое число **Максимум:** Ограничение на максимальное вводимое число. После ввода числа обязательно необходимо нажать клавишу **Enter** на клавиатуре.
Radio Group **радио группа IN** 	Создаёт на поле радио группу. В свойстве **Элементы** необходимо написать нужное число радио кнопок в формате **N□:□ИмяЭлемента**. Где N – целое число, которое будет отображаться на выходе блока, при выборе пункта NameGroup, а □ – знак пробела. Можно выделить только одно состояние на радио группе.
Check Group **группа флагов IN** 	Создаёт на поле группу флагов. В свойстве **Элементы** необходимо написать нужное число радио кнопок в формате **N□:□ИмяЭлемента**. Где N – целое число, а □ – знак пробела. На выходе будет отображаться результат побитовой операции ИЛИ из выделенных пунктов.
PROTO_IN блок считывания данных по адресу определённого протокола	См. раздел «Блоки организации коммуникации»
STRING_IN блок для ввода данных типа string (текстовые данные)	**Длина строки** – число символов для отображения **Смещение –** начало отображение символов

Отладка программы

Блок виртуального осциллографа

Осциллограф предназначен для графического отображения сигналов. Может работать в двух режимах: **по точкам** и **буфером**. Простой режим осциллографа предназначен для использования в режиме моделирования. Позволяет отображать на осциллографе заданное количество точек графика.

Буферный режим предназначен для использования в режиме обновления связи с контроллером, но может работать и в режиме моделирования. Позволяет отображать на осциллографе ограниченное число точек (ограничение вызвано областью памяти выделенной под данные в памяти контроллера).

Внимание: В режиме работы с чипом, после подключения/отключения сигналов к блоку осциллографа необходимо выполнить повторную загрузку конфигурационного файла в RAM (Нажать кнопку **Загрузить в ОЗУ**).

Настройка осциллографа

1. Откройте пример **Models_um_ScopeTest.mbp**. Для открытия примеров перейдите на вкладку **Помощь**, нажмите кнопку **Примеры**. Пример создан в библиотеке Models.
2. Зайдите в формулу **MAIN**. Собранная схема представлена на следующем рисунке:

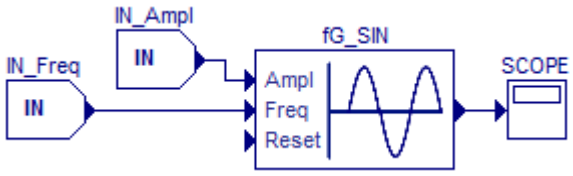


Рис. 110. Схема генератора синусоиды, подключенного к простому осциллографу

3. Два раза нажмите ЛКМ по блоку SCOPE. Откроется окно осциллографа:

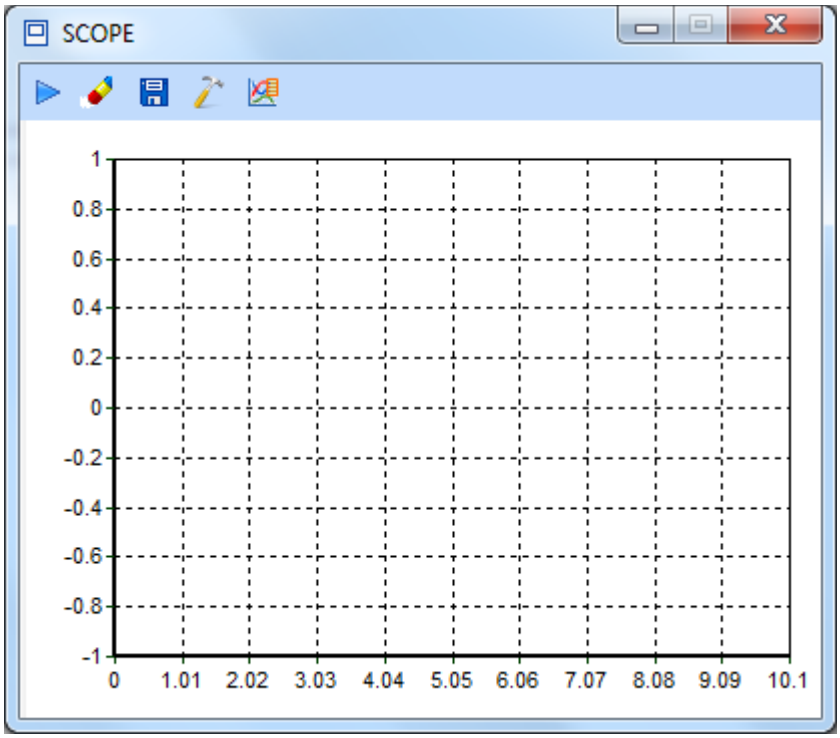


Рис. 111. Окно виртуального осциллографа

Назначение кнопок окна осциллографа:

Запуск/останов построения графиков осциллографом. Необходимо нажимать только для режима работы **Буфером**. В режиме работы **По точкам** обновление запустится автоматически., значок кнопки изменится на  . Если нажать ещё раз на кнопку, процесс построения графиков остановится и можно будет произвести сохранение графиков.

Очистить окно осциллографа от построенного графика. Если кнопка нажата в процессе моделирования, график начнёт строиться заново.

Сохранение графиков в формате рисунка *.bmp или в виде массива чисел. В сохранённом массиве первый столбец время, остальные – каналы по порядку с 1 по 6.

Кнопка вызова диалогового окна настроек осциллографа.

Кнопка вызова курсора, для просмотра значений графика.

4. Нажмите кнопку вызова настроек осциллографа  .

На рис. 112 представлены настройки осциллографа для библиотеки Models. Для других библиотек настройка **Количество точек** будет отличаться.

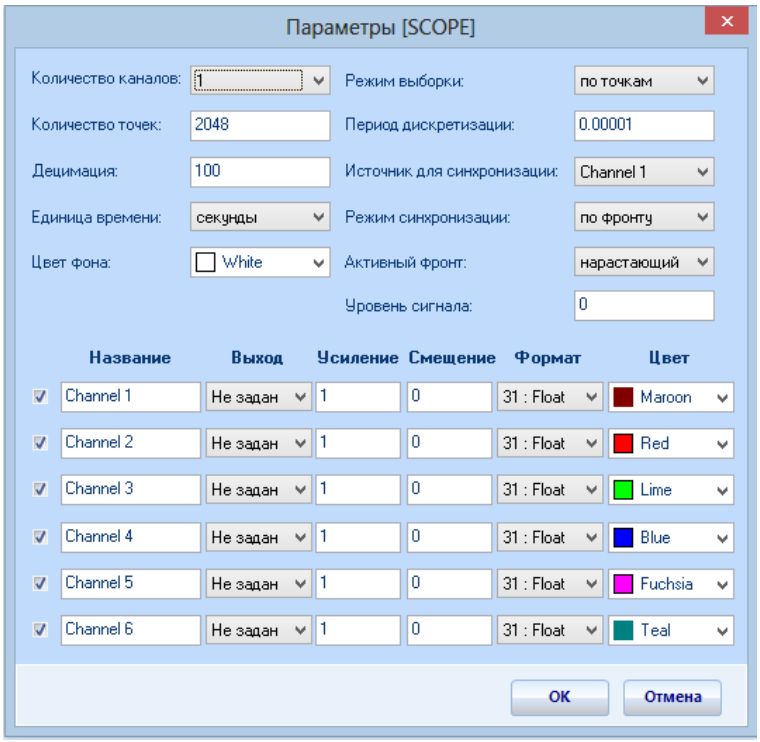


Рис. 112. Настройки осциллографа

Описание настроек осциллографа:

Количество каналов	Число каналов (входов) осциллографа. Количество каналов не более шести.
Количество точек	Число точек отображаемых на графике. При накоплении числа точек больше чем указано в данном параметре – график начнётся строиться заново. В режиме работы осциллографа **Буфер** число точек ограничивается буфером, выделенным под построение графиков. При добавлении дополнительного канала величина буфера поделится поровну между каналами. Подробней смотрите в **описании буферного режима работы осциллографа**.
Децимация	Настройка позволяющая проредить график, то есть отображать на графике каждую N точку. Таким образом, можно увеличить период снятия графиков в N раз. Если задан 0, то на графике строится каждая полученная точка.
Единица времени	Вид отображения времени. Можно выбрать: секунды, микросекунды, точки.
Цвет фона	Настройка цвета фона графика.
Режим выборки	Переключение режима осциллографа между **По точкам** и **Буфер**.

Период дискретизации	Дискретный шаг отображения графика. Должен быть равен частоте, на которой производится расчёт в данной FORMULA. В зависимости от шага дискретизации создаётся ось времени. **Количество точек**×**Период дискретизации**.

Настройка опции триггера

Источник для синхронизации	Список сигналов, из которых можно выбрать источник срабатывания триггера
Режим синхронизации	**авто** – автоматическое обновление графика, работа без триггера. **по фронту** – периодическое обновление графика по событию триггера. **однократный** – одиночное срабатывание обновления графика по событию триггера. Для повторного запуска нужно нажать кнопку  .
Активный фронт	**нарастающий** – срабатывание триггера по возрастающему фронту. **спадающий** – срабатывание триггера по спадающему фронту.
Уровень сигнала	Уровень срабатывания триггера.

Когда осциллограф работает опцией триггера, то в правом верхнем углу осциллографа отображается слово **Захват** – ожидание срабатывание триггера и считывание данных, **Готов** – построение графика.

Настройка отображения графиков:

Название	Название графика
Выход	Свободный выход осциллографа можно привязать к любому блоку OUT для построение графика.
Усиление	Коэффициент масштабирования графика
Смещение	Величина смещения графика
Формат	Формат данных сигнала графика.
Цвет	Цвет графика

5. Установите **Формат** сигналов, которые подключены к блоку осциллографа.
6. Нажмите на клавишу **OK** для применения настроек осциллографа.
7. Запустите симуляцию (кнопка **Начать моделирование** на вкладке **Домой** главного меню).
8. Наблюдайте процесс построения графика в окне осциллографа.

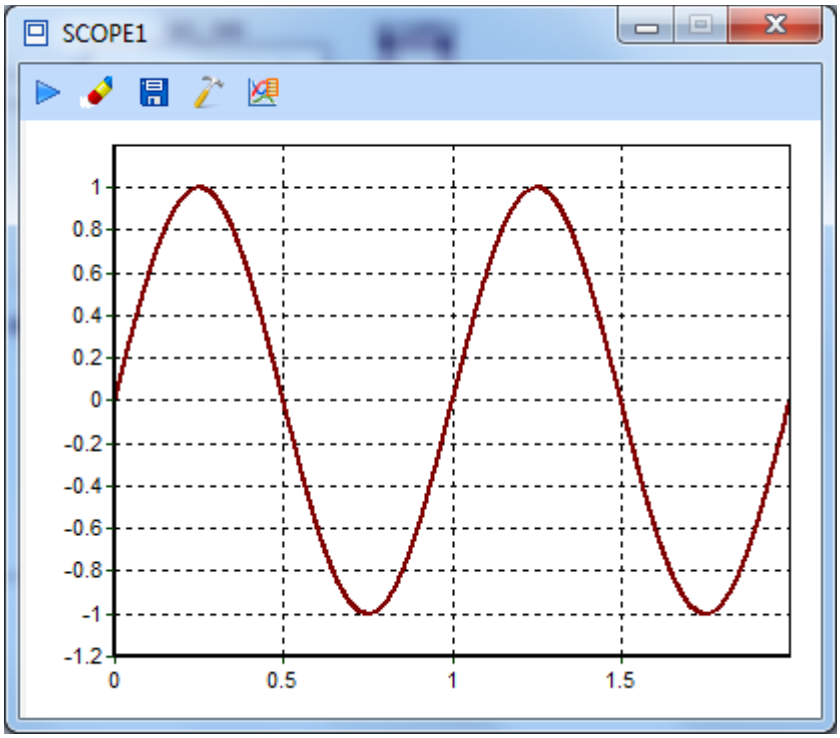


Рис. 113. Полученный график

Режим **По точкам** предназначен для потокового отображения сигнала на экране виртуального осциллографа. Режим не подходит для использования в режиме связи с контроллером (при отображении быстро изменяющихся сигналов во времени) из-за медленного процесса обмена данными между компьютером и микроконтроллером частота передачи сигнала не высока (период считывания порядка 0,7 секунды).

Режим триггера для режима По точкам

Для использования режима триггера необходимо:

1. Выбрать канал (опция **Источник для синхронизации**) по которому будет выполняться срабатывание триггера.
2. Выбрать режим (**Режим синхронизации)** как **по фронту** или **однократный.**
3. Выбрать срабатывания триггера (**Активный фронт**) **спадающий** или **нарастающий** сигнал.
4. Установить уровень срабатывания триггера – **Уровень сигнала**.
5. Запустить моделирование.

График будет отображаться при срабатывании триггера и не будет построен, пока не выполнится условие срабатывания триггера.

Режим Буфер работы осциллографа

В ядре **MexBIOS™** предусмотрена возможность просматривать в графическом виде изменение значений глобальных переменных. Чтобы построить график, изменения значения по тому или иному адресу глобальных данных, ядро **MexBIOS™** с заданной частотой сохраняет данные по этому адресу в массив. Такой массив вычитывается с заданной периодичностью (период равен времени заполнения буфера плюс время на передачу по каналу связи) из памяти микропроцессора. По переданному массиву, блок осциллографа, в буферном режиме работы, строит график. Процесс вычитывания данных отображается полосой загрузки под графиком осциллографа (в режиме обмена данными).

Одновременно ядро **MexBIOS™** может строить до 6 каналов. **MexBIOS™ Development Studio** выводит каналы в виде графиков.

Основное назначение буферного режима – получение графиков при работе с контроллером, но он может использоваться и в режиме моделирования.

После переключения простого осциллографа в буферный – значок осциллографа изменится, см. следующий рисунок:

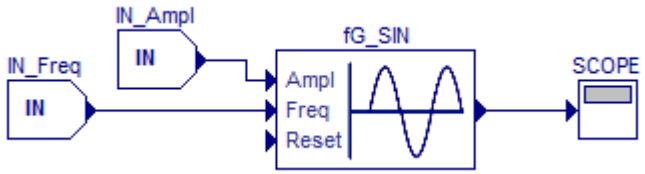


Рис. 114. Схема генератора синусоиды, подключенного к буферному осциллографу

Примечание: Осциллограф в буферном режиме может быть только один. При назначении другого осциллографа буферным, если был назначен ещё один буферный осциллограф, произойдёт сброс настроек для первого осциллографа в режим **по точкам**.

Пример работы с буферным осциллографом:

1. Откройте окно настройки осциллографа.
2. Измените режим выборки с **по точкам** на **буфером**.
3. Нажмите **OK**.
4. Повторно откройте окно опций осциллографа:

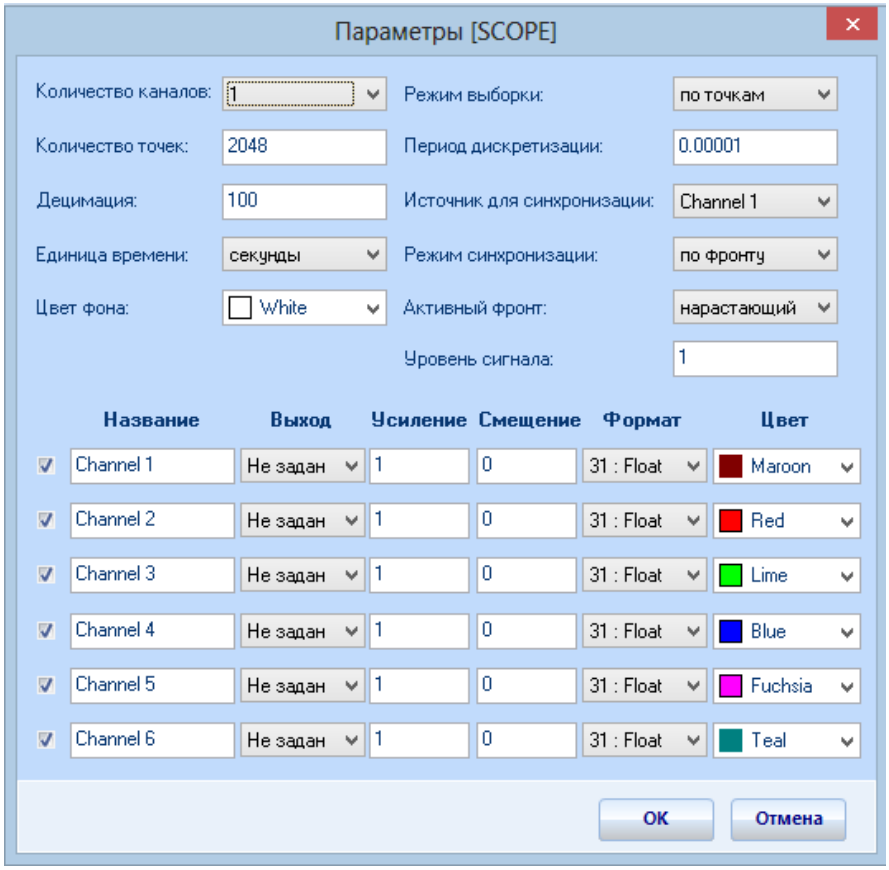


Рис. 115. Настройка «буферного» осциллографа

Величина **Количество точек** ограничилась значением **2048 (для библиотеки Models)**. При добавлении нового канала величина станет равной **1024**, это означает, что на построение одного графика будет отведено **1024** точек. Количество точек, отведённых на один канал, будет кратно уменьшаться при добавлении новых каналов.

5. Задайте параметр **Децимация** = 100. Это означает, что каждая 100 точка будет отображаться на графике, таким образом, график «сожмётся» по оси времени.
6. Закройте окно настроек, нажав кнопку OK.
7. Запустите процесс моделирования (кнопка **Начать моделирование** на вкладке **Домой** главного меню).
8. Построение графиков не начнётся автоматически при запуске моделирования. Необходимо нажать кнопку «пуск обновления данных» в окне SCOPE.
9. После запуска обновления данных, запустится накопление данных в массив.
10. При необходимости подстроить параметр **Децимация** (для сжатия графика по оси времени).
11. В режиме моделирования процесс накопления данных не будет показан. В режиме связи с микроконтроллером будет происходить отображение процесса накопления данных в массив.

Режим триггера для режима буфер

Для использования режима триггера в буферном осциллографе необходимо:

1. Выбрать канал (опция **Источник для синхронизации**) по которому будет выполняться срабатывание триггера.
2. Выбрать режим (**Режим синхронизации)** как **по фронту** или **однократный.**
3. Выбрать срабатывания триггера (**Активный фронт**) **спадающий** или **возрастающий** сигнал.
4. Установить уровень срабатывания триггера – Уровень сигнала.
5. Запустить моделирование или начать обновление.

График будет отображаться при срабатывании триггера и не будет построен, пока не выполнится условие срабатывания триггера.

Функция пошаговой отладки программы

В режиме моделирования реализована возможность пошаговой отладки программы. Данная функция позволяет пошагово отследить работу собранной схемы, корректность выполнения каждого блока. Режим доступен только в Моделировании.

Для того что бы перейти в режим пошагового моделирования необходимо нажать кнопку «Сделать шаг» на панели меню. Программа произведет расчет первого блока и перейдет к следующему, остановившись, в ожидании дальнейших действий. Отображение времени будет идти согласно заданному «периоду дискретизации».

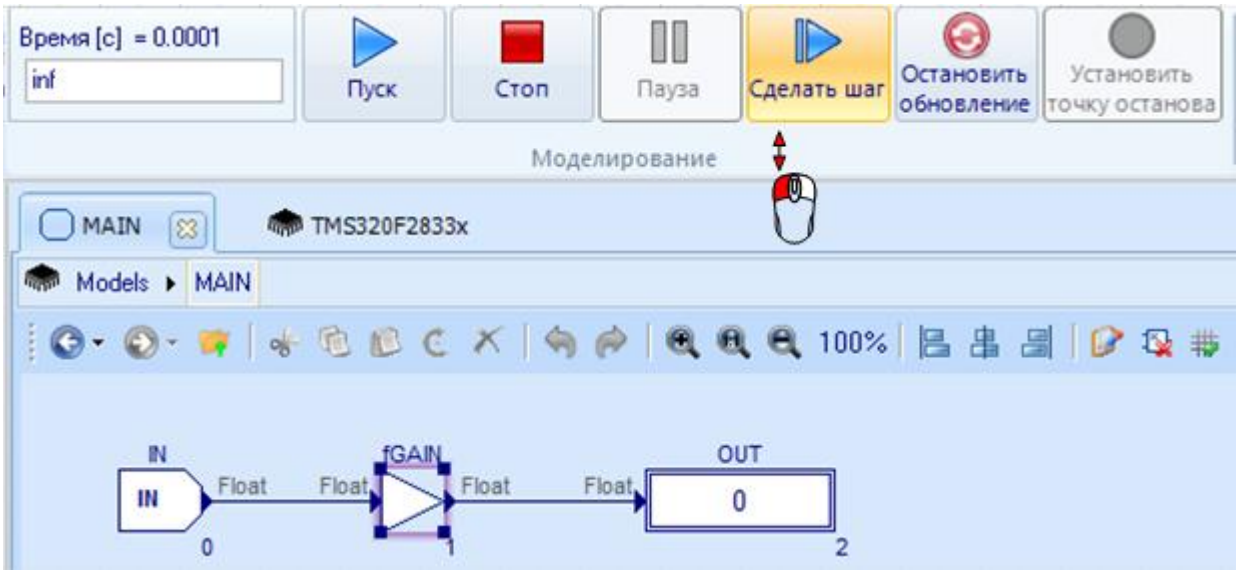
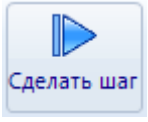
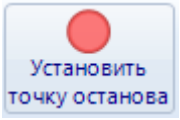
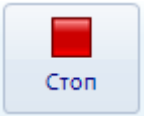


Рис. 116 Начало моделирования в режиме «Пошаговой отладки»

	Сделать шаг. При нажатии кнопки произойдёт вычисление следующего блока. Отладка не останавливается на IN, OUT и всех визуальных компонентах.

	Точку остановка можно назначить на блок, добавленный на поле набора из палитры. После установки точки останова можно нажать кнопку Пуск – программа произведёт цикл расчёта и остановится либо на следующей установленной точке останова, либо на данной точке останова.
	Кнопка стоп завершает процесс пошаговой отладки.

Функция пошаговой отладки **работает только в режиме Моделирование!**

Догрузка

Функция догрузки позволяет изменять программу в процессе работы на процессоре (либо в симуляции) без остановки выполнения программы.

После запуска обновления схемы, из палитры блоков на поле набора можно добавить любой другой блок. После добавления у такого блока появится значок плюса в правом верхнем углу, который означает, что блок не загружен в проект и его можно догрузить, в процессе работы, с помощью функции **Догрузка**.

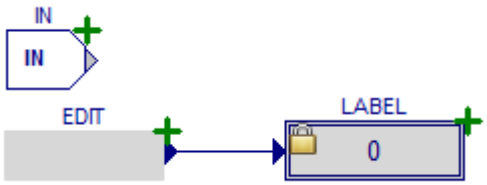


Рис. 117. Блоки готовые для добавления на исполнение в работающую программу

Также возможно осуществлять редактирование подключенных к блокам связей – удаление, переключение и добавление новых связей.

В **режиме моделирования** Догрузка осуществляется кнопкой **Догрузить** на вкладке **Домой**. После нажатия этой кнопки произойдёт догрузка в библиотеке Models и библиотеке процессора.

При **режиме обмена данными с устройством** догрузка в процессор осуществляется кнопкой **Догрузить** на вкладке **Устройство**, а в библиотеке Models догрузка осуществляется кнопкой Догрузить на вкладке Домой.

Ограничения функции Догрузка:

- Нельзя удалять исполняющиеся в процессоре блоки.
- Нельзя добавлять элементы алгоритмов (блоки EVENT, IF, WHILE, STATE_FLOW, ALGORITHM).

Выполнение Догрузки в процессе запущенного ШИМ и работы силовой части может привести к выходу оборудования из строя.

Работа с библиотеками блоков

Работа с библиотеками блоков предполагает: создание новых блоков, удаление блоков, редактирование существующих блоков, скрытие исходного кода блоков. Пользователь может создавать собственные блоки с помощью **Редактора блока**.

Исходный код блока может быть открыт или закрыт для редактирования. Закрытие библиотеки необходимо для скрытия исходного кода от пользователя, в библиотеку встраиваются только откомпилированные блоки и готовый out файл библиотеки для самостоятельной загрузки в flash-память. При этом пользователь может создавать собственные блоки и размещать их в отдельную секцию flash-памяти.

При открытой библиотеке все блоки доступны для редактирования с помощью инструмента редактирования блока. Пользователь может по своему усмотрению вносить изменения и дорабатывать блоки, предложенные в библиотеке. Вся ответственность при таком редактировании ложится на пользователя.

Рекомендуется не редактировать блоки среды, а создавать копии существующих блоков. Для создания копии нужно в **Панели элементов** нажать ПКМ по нужному блоку и выбрать пункт **Копия**.

Редактор блока вызывается отдельно для каждого блока двойным нажатием левой клавишей мыши по блоку в **Панели элементов**. Также конструктор можно вызвать, нажав на блок правой клавишей мыши и в ниспадающем меню выбрать пункт **Редактировать код**. Изменения затронут все экземпляры блока, на открытой схеме.

Внимание: После редактирования исходного кода блока и компиляции все настройки в текущей схеме собьются на указанные в **Редакторе блока**.

Редактор блока

Особенностью создания и редактирования блоков является назначение уникального номера **ID**, состоящего из уникального набора байтов. Для осуществления проверки соответствия блоков загруженных в чип и находящихся в библиотеке необходимо на вкладке **Правка** нажать кнопку **Проверить блоки**. После завершения процедуры проверки, появится сообщение об удачной верификации библиотеки или перечень блоков, **ID** которых не совпал с номером **ID** в текущей библиотеке. Если такие блоки есть, то произойдёт блокирование их для выноса на поле набора.

Окно конструктора блоков представлено на следующем рисунке:

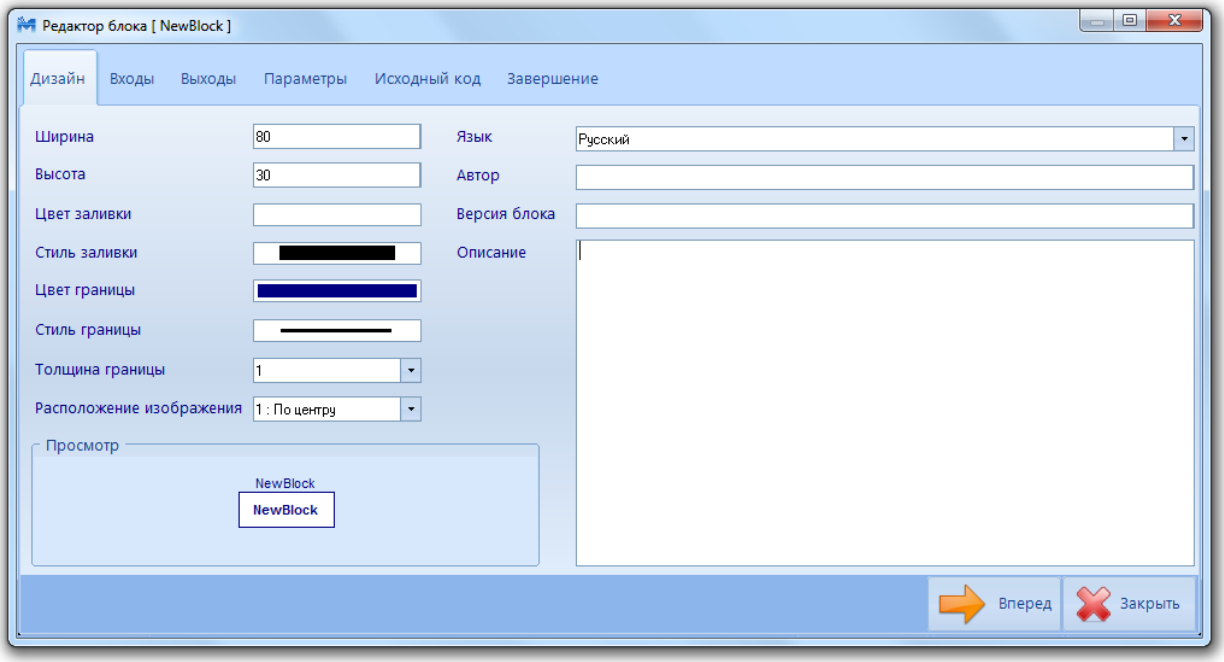


Рис. 118. Диалоговое окно конструктора блоков

Создание и редактирование блоков

Создание блока представляет собой последовательность из нескольких шагов, которые производятся в каждой отдельной вкладке конструктора. Переключение между вкладками конструктора блока происходит непосредственным нажатием на соответствующую вкладку либо на кнопки **Назад** / **Вперед** в нижней панели. Кнопка **Закрыть** закрывает конструктор блока.

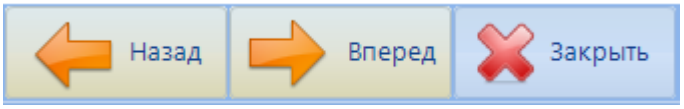


Рис. 119. Кнопки навигации

Закладка Входы

Закладка **Входы** предназначена для создания входов блока и настройки формата данных.

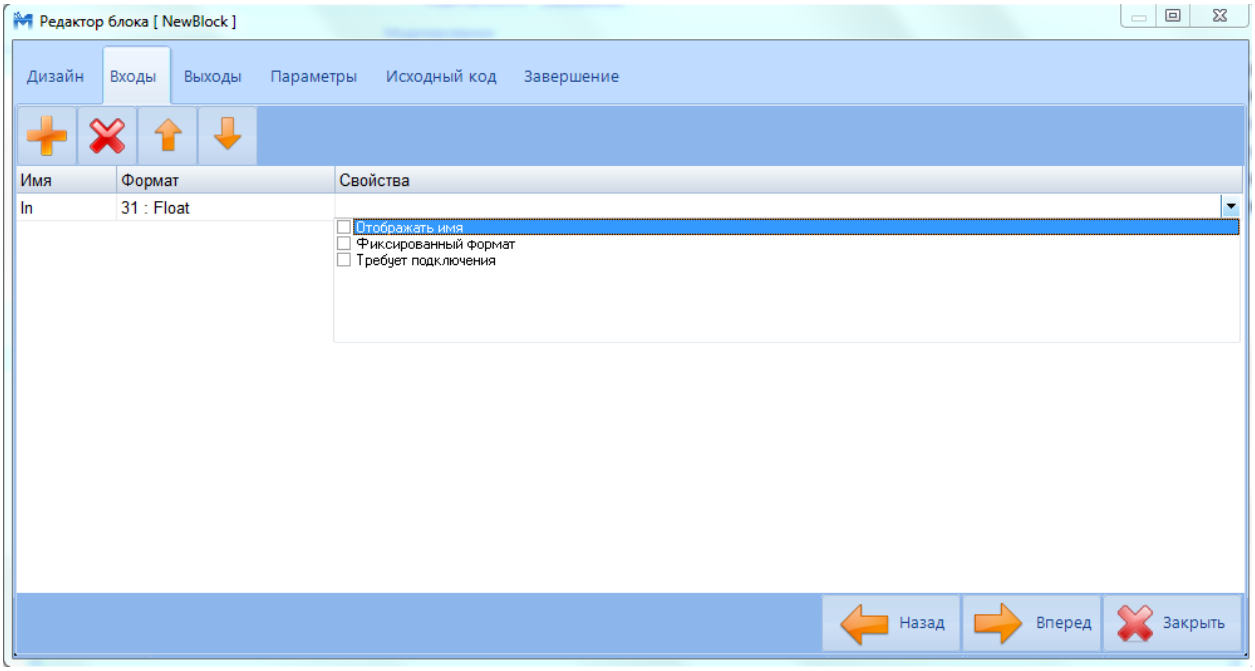


Рис. 120. Закладка Входы

Описание редактируемых параметров закладки **Входы** приведено ниже.

- **Имя** – имя входа, отображаемого на блоке. Заданное имя входа используется в Code editor, для обращения к данным поступающим через вход и обработки их внутри исполняемого кода блока. Внутри кода блока не допускается присвоение нового значения входа.
- **Формат** — формат данных входа, назначается из ниспадающего списка форматов. Список вызывается левым щелчком мыши, после выделение ячейки.

  Параметр **Формат** определяет способ хранения и передачи данных между блоками в ядре операционной среды. Существует три основных варианта представления данных:

  1) Целочисленное значение типа **0: Integer**. Например, значение «5» будет представлено как «5» в десятичном формате.
  2) Дробное значение с фиксированной запятой типа **n: Qn** (**n** – определяет степень формата). Степень «**n**» определяет диапазон возможных значений и точность хранения дробной части числа представленной в дополнительном коде. Например, значение «5.0» в формате «**Q24**» будет представлено как «**5.0 * 2^{24}**» = «**83886080**» в десятичном формате или «**0x5000000**» в шестнадцатеричном формате. При этом для приведенного формата под дробную часть отведено 24 бита, под целую 31-27=7 бит, а возможный диапазон значений будет от «-128.0» до «127.(9)».
  3) Дробное значение с плавающей запятой типа **31: Float**. Например, значение «5» будет представлено как «**1084227584**» в десятичном формате или «**0x40A00000**» в шестнадцатеричном формате.

Примечание: Для преобразования значений типа «**31: Float**» в целое и обратно необходимо на схеме ставить блоки преобразований формата.

- Для каждого входа можно назначить дополнительные свойства:

  Отображать имя – отображать имя входа на блоке или нет.
  Фиксированный формат – свойство для отображения формата входа. При изменяемом формате входа необходимо убрать это свойство.
  Требует подключения – если отмечено это свойство, то в панели Ошибки и Предупреждения будет выдавать сообщение, что порт данного блока не подключен.

  Для работы с таблицей контактов в верхней части закладки имеется панель с кнопками.

Рис. 121. Кнопки работы с таблицей контактов

Обозначения кнопок:
1 – добавление нового контакта;
2 – удалить выделенный контакт из таблицы;
3 – двигать выделенный контакт вверх по списку;
4 – двигать выделенный контакт вниз по списку.

Закладка Выходы

Закладка **Выходы** предназначена для настройки выходных параметров блока.

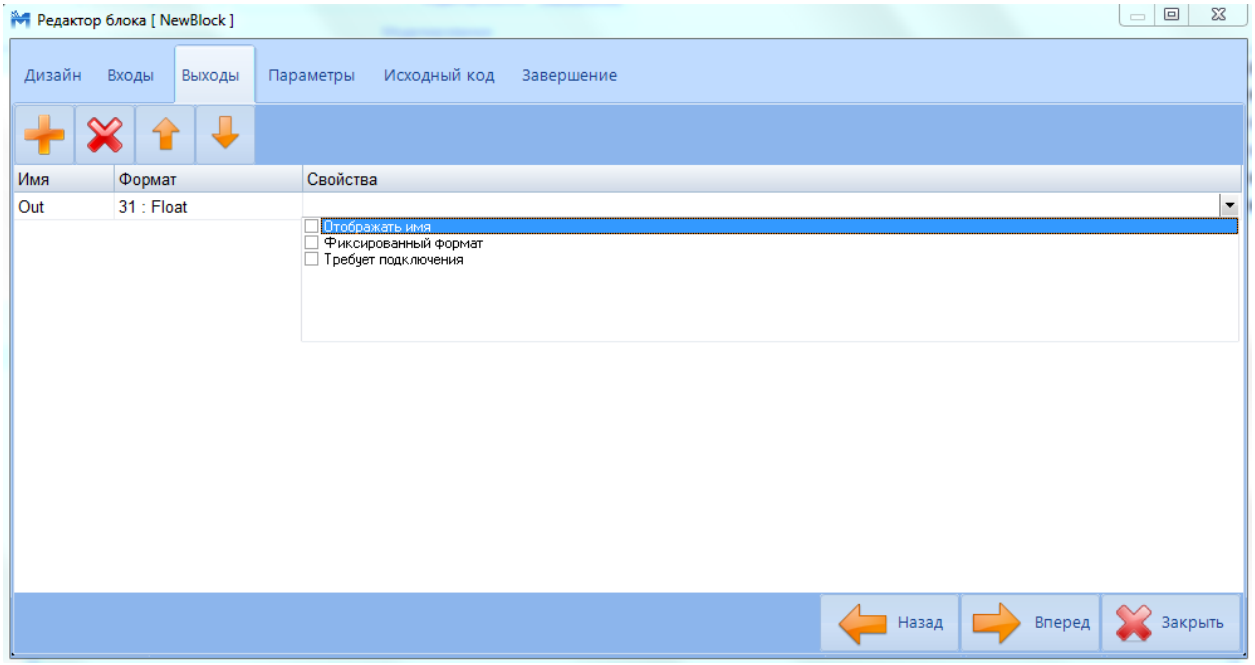


Рис. 122. Закладка Выходы

Диалог идентичен диалогу вкладки **Входы** и имеет те же функции. Отличие лишь в создании выхода блока. Внешний вид блока можно посмотреть на вкладке **Дизайн**.

Закладка Параметры

Закладка **Параметры** предназначена для создания и настройки внутренних параметров и переменных в функции блока, написанной в **Редакторе блока**.

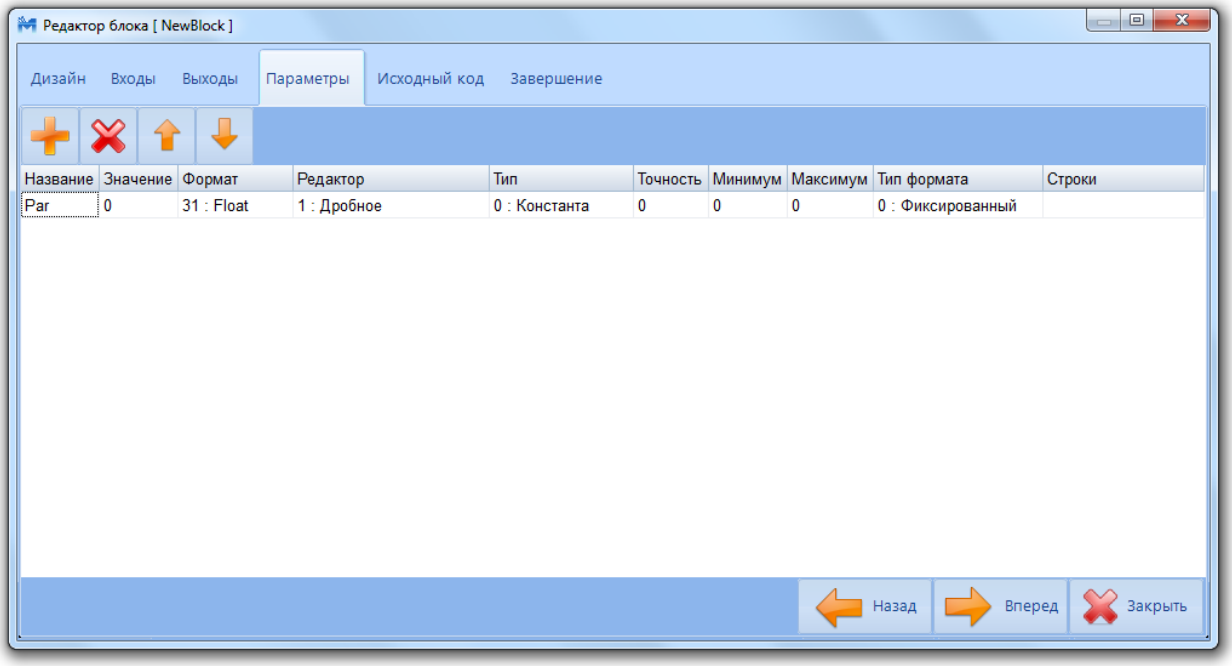


Рис. 123. Закладка **Параметры**

Описание редактируемых параметров закладки **Параметры** приведено ниже.

- **Название** – Имя параметра, к которому происходит обращение в **Редакторе блока**.

  Определяет, как будет отображаться имя параметра на панели **Свойства** и в окне **Параметры** среды **MexBIOS™ Development Studio**.

- **Значение** – Начальное значение параметра. Определяет значение, которое будет задано в параметре в момент инициализации блока.

- **Формат** – Формат данных параметра. Аналогично полю **Формат** во вкладке **Входы** и **Выходы**.

Формат **32: Double** возможно применять только для моделирования!

- **Редактор** – Тип редактирования параметра в **Свойствах**. Определяет способ отображения и ввода значений параметров блока на панели **Свойства** среды **MexBIOS™ Development Studio**. Существуют следующие типы:

  1) **Целое** – целочисленные значения (допускается ввод с клавиатуры цифр от «0» до «9» и знака «-»).
  2) **Дробное** – дробные значения (допускается ввод с клавиатуры цифр от «0» до «9», а также знаков «-» и «.»).
  3) **Булево** – выпадающий список (возможные значения выбираются из списка содержащих две строки: «**0: Нет**» и «**1: Да**»).
  4) **Шестнадцатеричное** – шестнадцатеричный редактор (допускается ввод с клавиатуры символов от «0» до «9» и символов от «A» до «F»).
  5) **Выпадающий список** – выпадающий список (возможные значения выбираются из

списка сформированного пользователем).

6) **Вход/Выход** – выпадающий список (возможные значения выбираются из списка содержащих две строки: «**0: Вход**» и «**1: Выход**»).
7) **Количество входов** – целочисленные значения (определяет количество входных контактов блока).
8) **Количество выходов** – целочисленные значения (определяет количество выходных контактов блока).
9) **Формат** – выпадающий список (возможные значения выбираются из списка возможных форматов).
10) **Адрес** –
11) **Списки Gpio, Spi, Sci, Pwm1, Pwm2, Adc, Qep** – содержат списки доступных

- **Тип** – Тип параметра.

Возможны три следующих типа:

1) **Константа** – значение параметра есть константа, которая может задаваться только во время инициализации (до запуска моделирования или обмена данными с процессором). Редактируется только через «Свойства».

2) **Переменная** – значение параметра есть переменная, которая может задаваться как во время инициализации, так и во время симуляции и работы с процессором. Редактируется через **Свойства** до запуска процесса моделирования или режима работы с процессором и в окно **Переменные** после добавления параметров блока в окно **Переменные** и запуска симуляции или режима работы с процессором.

3) **История** – значение параметра есть переменная, которая используется для работы блока. Данный параметр недоступен для редактирования. Параметр данного типа позволяет сохранять значение переменной после каждого цикла исполнения функции блока.

- **Точность** – Точность задаваемого значения.

Внимание: Необходимо задать параметр **Точность** для параметра **Дробное**. Иначе нельзя будет задавать дробную часть. Параметр **Точность** определяет количество знаков после запятой отображаемых в окне **Свойства** и **Переменные**.

- **Минимум** – Минимальное значение параметра.

Определяет минимально возможное вводимое значение.

- **Максимум** – Максимальное значение параметра

Определяет максимально возможное вводимое значение.

Примечание: Если «Минимум = 0» и «Максимум = 0», то используется весь диапазон значений для заданного формата.

- **Тип формата** – фиксированный или зависимый параметр.
- **Строки** – содержит сформированный пользователем список строк, выпадающий при вводе значения параметров, если задан тип редактора «Строки».

Вкладка Исходный код

Предназначена для редактирования исходного кода блока.

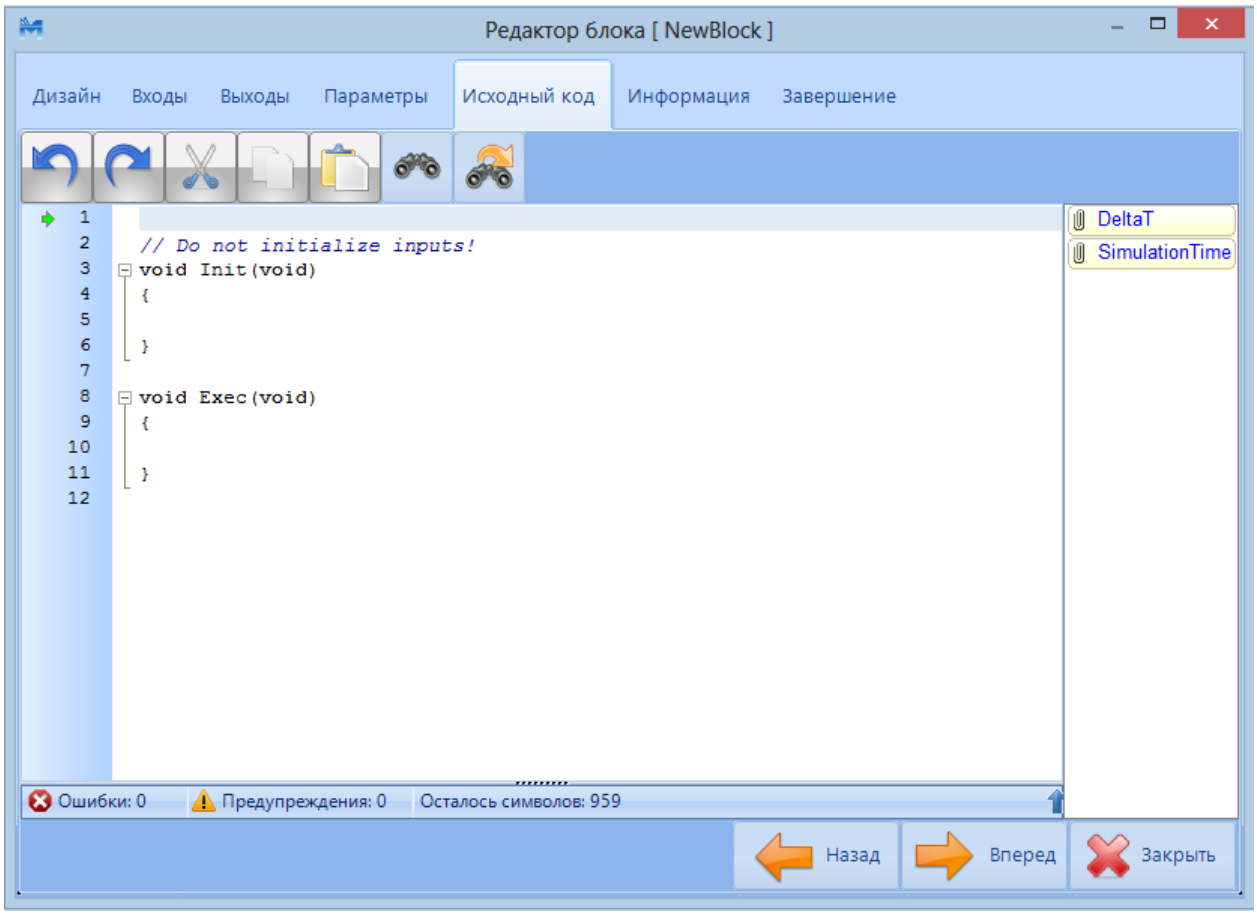


Рис. 124. Закладка Исходный код

Содержит две обязательные функции:
Init – для инициализации выходных данных и параметров.

Exec – для описания основной функции обработки данных.

Для удобства работы в правой верней части редактора, расположен список всех входов/выходов и параметров блока. Для добавления в программный код имен переменных и параметров просто перенесите с помощью мыши выбранное имя на поле редактирования кода,

после чего оно появится в том месте, где был расположен курсор ввода с клавиатуры.

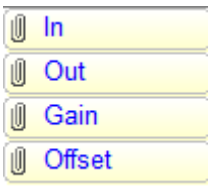


Рис. 125. Панель списков входов/выходов и параметров блока

Для работы с кодом в верхней части закладки имеется панель с кнопками.

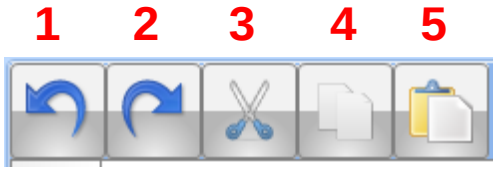


Рис. 126. Кнопки управления кодом

1 – Отмена предыдущего действия редактирования текста.
2 – Вернуть отмененное действие редактирования текста.
3 – Вырезать выделенный текст.
4 – Копировать выделенный текст.
5 – Вставить из буфера обмена текст.

Внизу располагается панель информационная панель. При компиляции в информационную панель будут выводиться сообщения об ошибках и предупреждения.

Также в информационной панели выводиться оставшееся число символов, которое может быть использовано в демонстрационной версии программы.

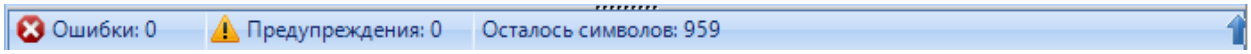


Рис. 127. Информационная панель

Вкладка Завершение

Внешний вид вкладки представлен на следующем рисунке:

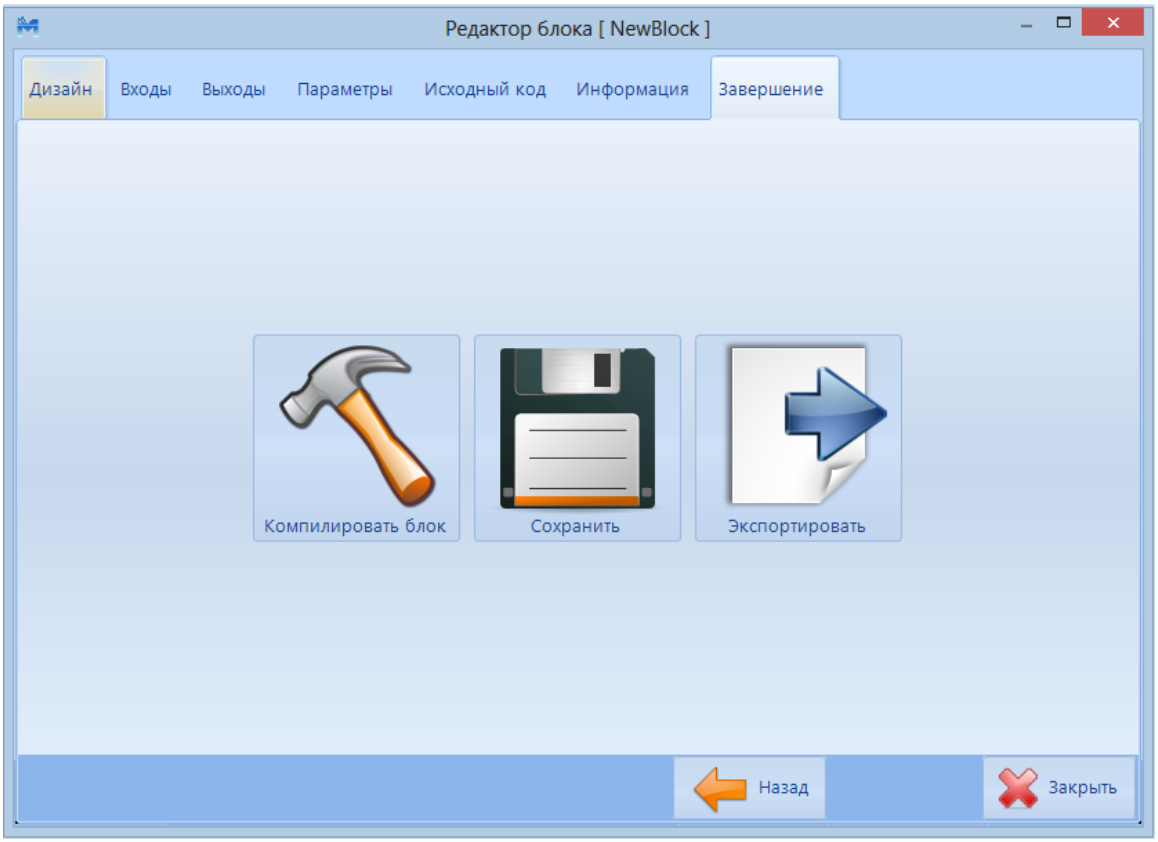


Рис. 128. Вкладка **Завершение**

Вкладка предназначена для завершающих действий над созданным блоком. Предоставляется произвести компиляцию блока кнопкой **Компилировать блок**. Если блок не завершен, можно сохранить произведённые изменения кнопкой **Сохранить**. Для экспорта созданного блока необходимо нажать кнопку **Экспортировать** и задать директорию, куда будет скопирована папка с исходными файлами блока.

Создание собственного блока

Блок можно создать в уже существующей группе блоков, либо создать новую группу блоков. Можно дублировать существующий блок, отредактировать для решения своих задач.

Назначение кнопок в контекстном меню палитры:

Для появления контекстного меню палитры необходимо нажать ПКМ по группе либо блоку. Вид меню представлен на следующем рисунке:

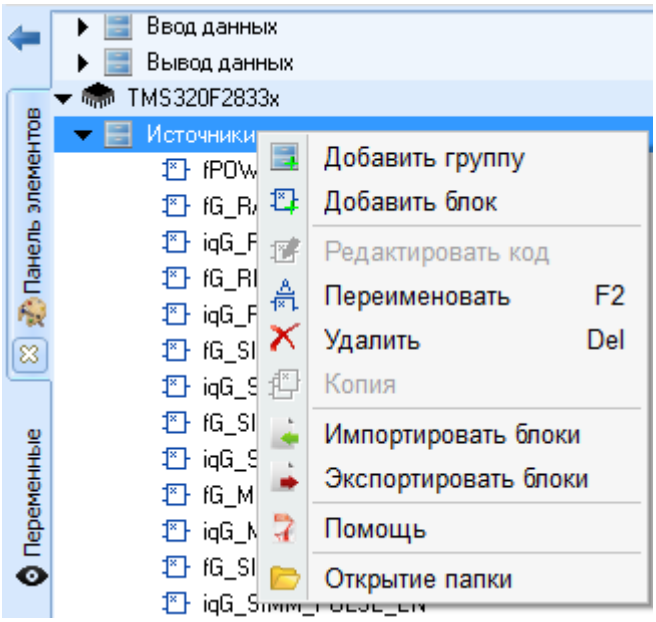


Рис. 129. Контекстное меню в палитре

Добавить группу – создать новую группу в палитре;
Добавить блок – добавить новый блок в текущую группу (которая выделена или в которой находится выделенный блок);
Редактировать код – вызов Редактора кода для редактирования выделенного блока.
Переименовать – переименовать группу либо блок, на данную функцию назначена горячая клавиша **F2**. Чтобы переименовать группу или блок, необходимо выделить группу/блок и нажать **F2** либо ПКМ в контекстном меню выбрать пункт **Переименовать**.

Кнопка Delete не работает при редактировании имени блока. Используете BackSpace.

Удалить – удаляет группу/блок из палитры. Возможно удалить блок из палитры и с диска (полное удаление) либо удалить из палитры, тогда блок можно будет вернуть через меню **Параметры/Библиотека** вкладка **Блоки**. После нажатия **del** появляется окно с требованием подтвердить действие, если нажать **Да** – произойдёт полное удаление файла, **Нет** – файл удалится из палитры блоков, **Отмена** – отмена удаления.

Блок или группа удалится безвозвратно с жесткого диска компьютера при полном удалении

Копия – создаёт копию выделенного блока в текущей группе.
Импортировать блоки – импортирует блок из указанной папки. Блок добавляется в текущую папку. После нажатия пункта **Импортировать блоки** появится диалог выбора каталога с блоком/блоками. Если в выбранном каталоге находится несколько блоков, то произойдёт добавление всех блоков в текущую группу.
Экспортировать блоки – экспортирует выделенный блок или выделенную группу в указанную папку.
Помощь – вызов справки по блоку.
Открытие папки – открыть папку с файлами выделенного блока.

Рекомендуем не добавлять новые блоки в существующие группы. Создавайте новые группы.

1. Создание нового блока. В качестве примера создадим блок сумматора в библиотеке Models:
   1.1 Нажать ПКМ по названию библиотеки.
   1.2 В контекстном меню выбрать пункт **Добавить группу**.

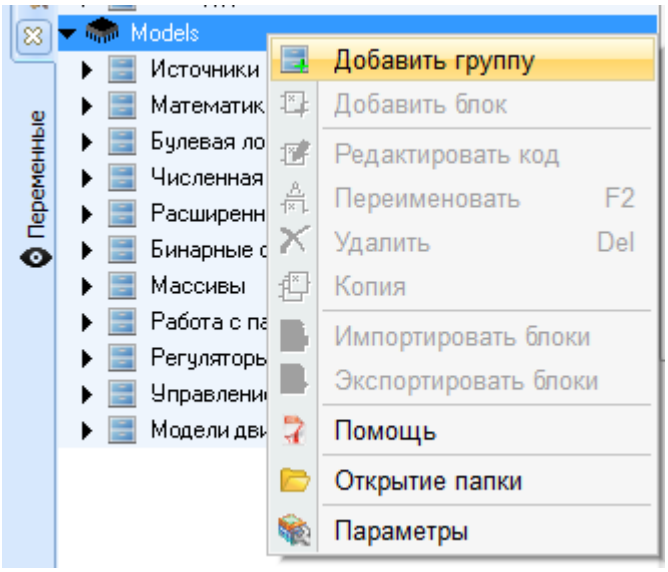


Рис. 130. Добавление группы в библиотеку

   1.3 В конце списка групп появится новая группа с именем **NewGroup**.

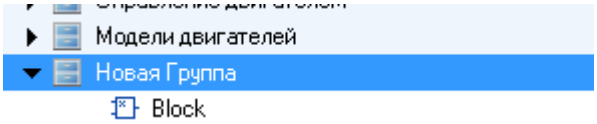


Рис. 131. Созданная новая группа

   1.4 Нажать ПКМ по группе. Выбрать пункт **Добавить блок**
   1.5 В группе появится новый блок.
2. Переименование блока:
   2.1 Нажать ПКМ по имени нового блока
   2.2 В контекстном меню выбрать **Переименовать**.

2.3 Либо выделить блок и нажать F2.
2.4 Переименуйте новый блок, например **myADD**
3. Два раза нажмите ЛКМ по блоку либо вызовите контекстное меню и нажмите **Редактировать**. Появится вкладка **Редактор блока**.

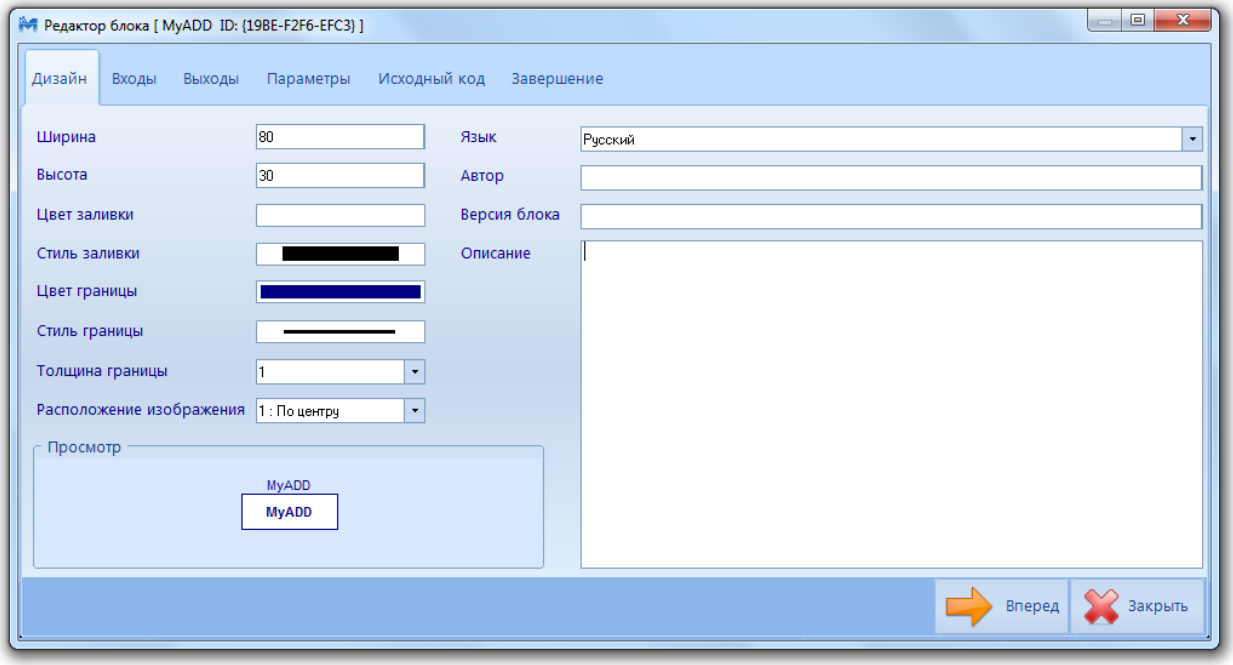


Рис. 132. Окно Редактор блока

О настройках блока в **Редактора блока** смотрите в разделе «Конструктор блока».

4. На вкладке **Дизайн** отредактируйте внешний вид блока.
5. Перейдите на вкладку **Входы** Добавьте два входа. Задайте формат **Integer**.

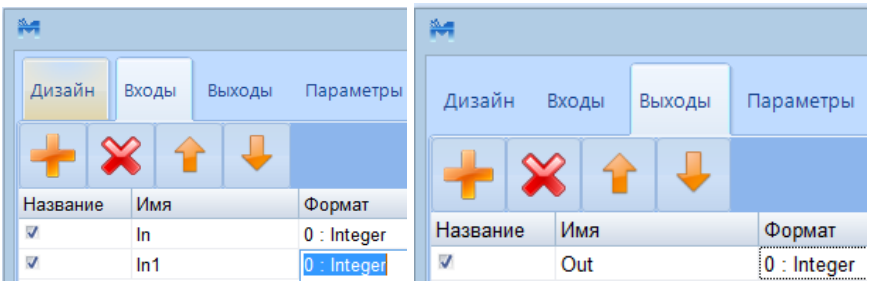


Рис. 133. Вкладка Input и Output

6. Перейдите на вкладку **Выходы** Добавьте один выход. Задайте формат **Integer**.
7. В блоке сумматора нет параметров.
8. Перейдите на вкладку **Исходный код**.
9. Введите следующий код:

```
void Init(void)
{
}
void Exec(void)
{
v->Out = *v->In + *v->In1;
}
```

На выход блока будет поступать сумма двух входных сигналов.

При написании кода следует руководствоваться следующими правилами:

- Обращение к любому входу блока должно происходить через указатель;
- Использования любого элемента описанного ранее (как во вкладках Входы, Выходы, Переменные, так и к другим блокам) должно происходить с использованием обращения к элементу структуры блока через конструкцию v->.

10. Если есть необходимость - задайте информацию о блоке на вкладке **Информация**.
11. Перейдите на вкладку **Дизайн**. Убедитесь, что внешний вид блока удовлетворителен.
12. Перейдите на вкладку **Завершение**. Нажмите кнопку **Компилировать блок**. Если компиляция прошла успешно, то на окне отображения процесса компиляции появится сообщение **Компиляция завершена**.
13. Если в коде имеются ошибки, то на вкладке **Исходный код** появится панель отображения ошибок и предупреждений:

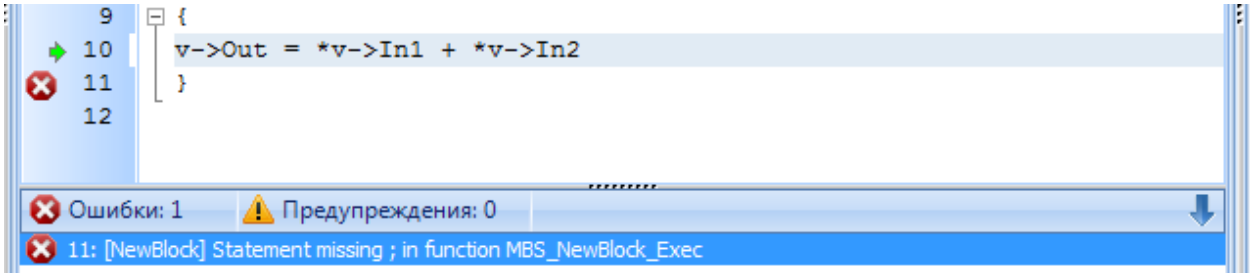


Рис. 134. Панель отображения ошибок и предупреждений

14. Если компиляция прошла успешно, блок создан.
15. Закройте окно **Редактор блока**.
16. При необходимости переместите блок в секцию памяти блоков пользователя. См. раздел «**Библиотека блоков пользователя**».
17. Нажмите кнопку компиляции библиотеки **Построить** на вкладке **Правка**. Произойдёт компиляция библиотеки. В папке библиотеки обновится *.out и *.hex файлы, которые нужно загрузить в контроллер. По загрузке hex файла в память контроллера см. раздел «**Загрузка библиотеки и стартового проекта в память контроллера**».
18. Перед загрузкой проверьте работу блока в режиме симуляции. При необходимости провести редактирование блока.

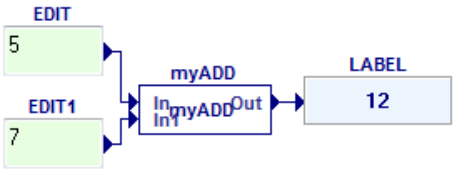


Рис. 135. Работа нового блока

Настройки компиляции библиотек

Для компиляции внешнего блока и затем библиотеки необходимо чтобы был установлен компилятор, предоставляемый фирмой производителем используемой серии микропроцессора. Настройка расположения компилятора (выбор папки, где находится компилятор) производится через диалог **Библиотека**, вкладка **Параметры построения**, который вызывается кнопкой **Настройки** в меню **Правка**.

Внимание: Неправильное задание настроек в окне **Настройки** может привести к потере связи с контроллером.

Вкладка Библиотека

При добавлении блока или изменении существующего в среде **MexBIOS™ Development Studio** блока необходимо перестроить (провести компиляцию) соответствующую библиотеку для последующей симуляции и/или загрузки в память устройства. Кроме того, при использовании стартового проекта, отличного от поставляемого при установке библиотеки, также необходимо провести его компиляцию. Для этой цели в среде предусмотрено ряд настроек расположенных на следующих вкладках:

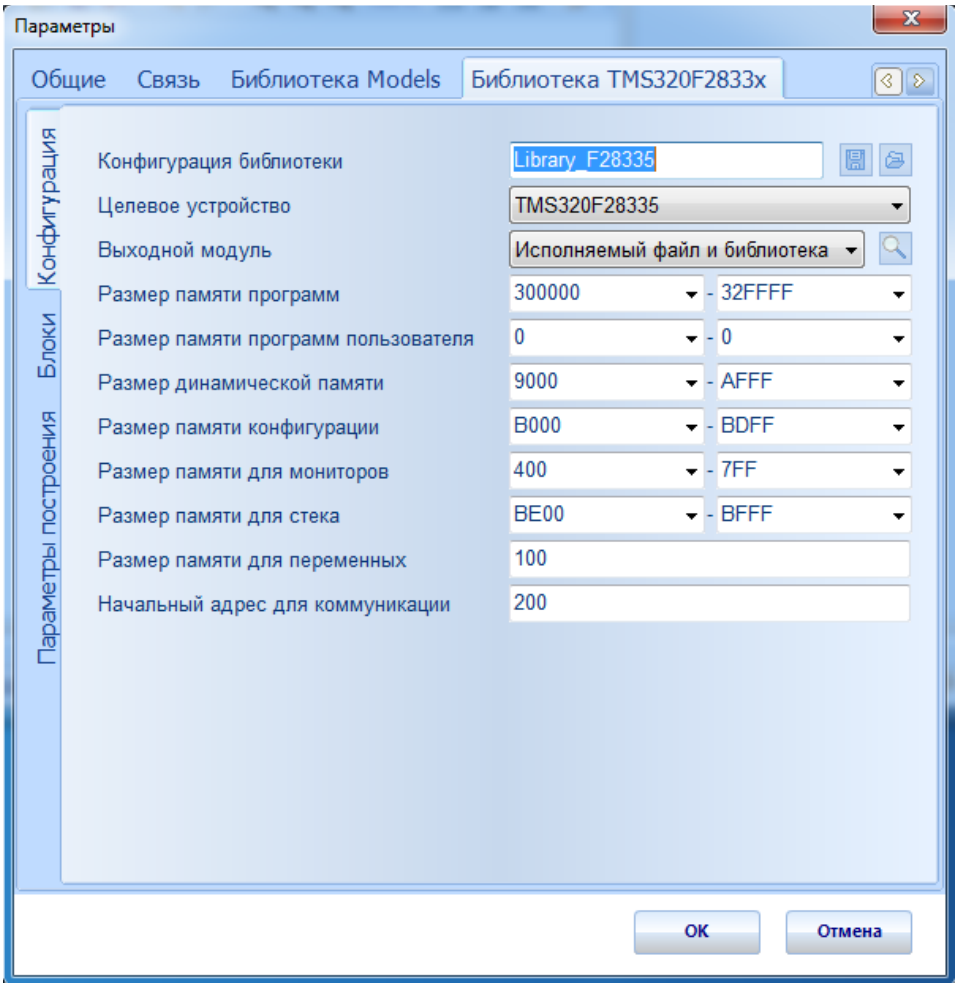


Рис. 136. Пример окна Конфигурация вкладки Библиотека

Вкладка **Конфигурация** – содержит базовые настройки библиотеки, которые определяют расположение в памяти устройства **MexBIOS™ kernel**, тип используемого контроллера, тип выходного файла библиотеки и параметры для коммуникации с библиотекой.

Внимание: Неправильное задание настроек может привести к потере связи с устройством.

Конфигурация библиотеки – отражает имя файла конфигурации, содержащие все настройки для библиотеки.

После изменения каких-либо настроек во вкладке **Library** существует возможность сохранить их в отдельном файле конфигурации с расширением «.mbcfg». Для этого необходимо нажать на кнопку и задать имя нового файла. При этом файлы, поставляемые при установке библиотеки, изменять не рекомендуется (они помечены атрибутом файла «Только для чтения» и недоступны для изменения).

Для загрузки ранее сохраненной конфигурации необходимо нажать на кнопку и выбрать файл с расширением «.mbcfg». После этого в поле «Конфигурация библиотеки» отобразится имя файла без расширения.

Целевое устройство – тип используемого контроллера.

Изменение данной настройки приводит к изменению секторов памяти отображаемых в выпадающих списках.

Выходной модуль – тип выходного файла.

Доступны следующие значения: «Исполняемый файл» (только исполняемый файл), «Статическая библиотека» (только статическая библиотека), «Исполняемый файл и библиотека» (исполняемый файл и статическая библиотека). Для просмотра папки хранения выходных фалов необходимо нажать на кнопку  .

Размер памяти программ – диапазон адресов в памяти программ (в формате Hex) для хранения исполняемого кода основной сборки библиотеки.

Размер памяти программ пользователя – диапазон адресов в памяти программ (в формате Hex) для хранения дополнительной пользовательской сборки библиотеки.

Размер динамической памяти – диапазон адресов в памяти данных (в формате Hex) для динамически создаваемых объектов

Размер памяти конфигураций – диапазон адресов в памяти данных (в формате Hex) для загрузки и хранения конфигурации работы библиотеки

Размер памяти для мониторов– диапазон адресов в памяти данных (в формате Hex) для мониторинга данных в режиме реального времени (в среде используется для блока **SCOPE** в режиме «Буфер»)

Размер памяти для стека – диапазон адресов в памяти данных (в формате Hex) для внутренних переменных и стека

Размер памяти для переменных – размер памяти в словах (в формате Hex) выделенных под переменные среды. Например, при значении 100 (Hex) = 256 (dec) доступно использовать 256 / 2 = 128 переменных (включая не только блоки VAR, но и переменные блоков библиотеки)

Начальный адрес для коммуникации – начальный адрес в протокола обмена. Все выше описанные секции размещаются начиная с заданного адреса в соответствии с их размером.

Предопределённая конфигурация – предустановленная конфигурация работы библиотеки, которая будет использоваться при старте микроконтроллера. Данная опция может использоваться при отсутствии внешней памяти для хранения конфигурации.

Внимание: Данная опция отсутствует в демонстрационной версии среды.

Вкладка **Блоки** – включает дополнительные настройки расположения блоков библиотеки. С помощью данного окна можно запретить использование выбранных блоков и перенести выбранные блоки в секцию пользователя или исключить их из сборки.

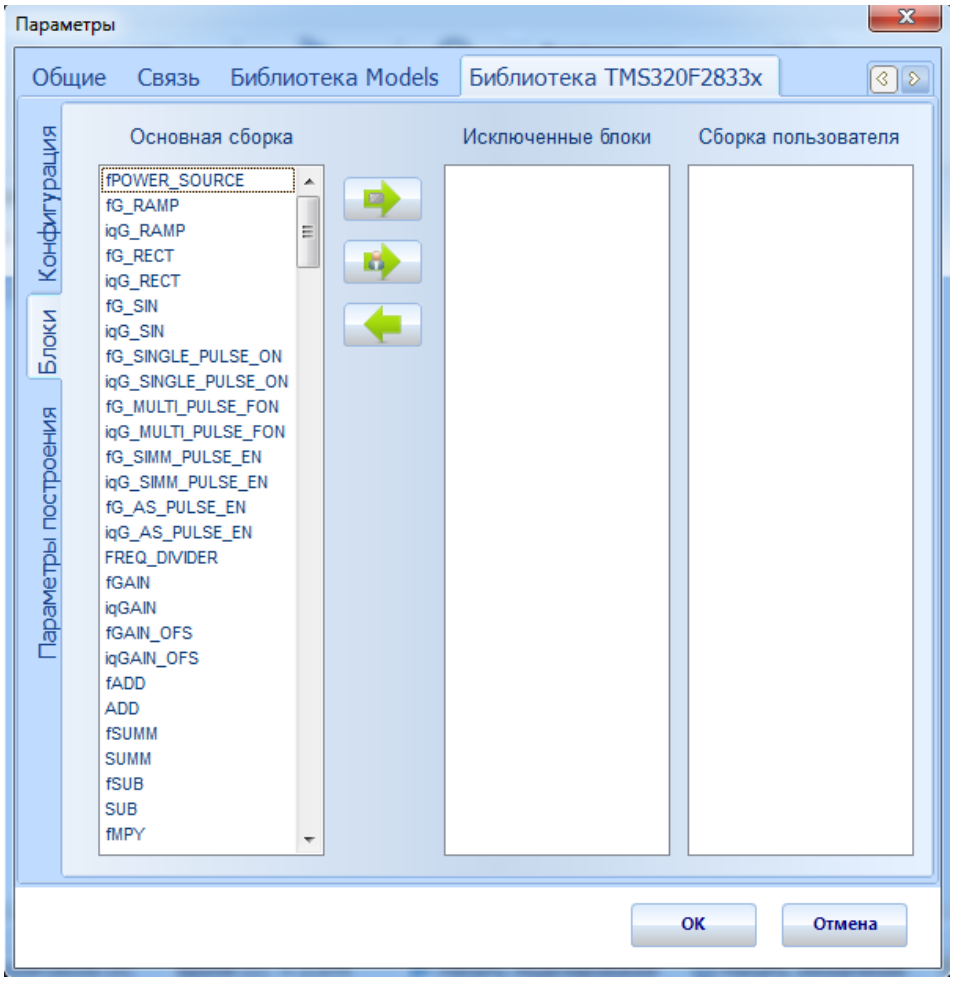


Рис. 137. Пример окна Блоки вкладки Библиотека

Для запрета использования блока необходимо:

- В списке «**Основная сборка**» выделить нужный блок.
- Нажать кнопку  .
- Блок появится в списке «Исключённые блоки». После нажатия кнопки «OK», в палитре возле блока появится серый прямоугольник – знак, что блок запрещён к использованию.

Секция пользователя – отдельно выделенная память в контроллере, где пользователь может хранить свои блоки отдельно от блоков библиотеки. Для секции пользователя генерируется отдельный выходной файл, который необходимо заносить в память устройства отдельно.

Для переноса блока в секцию пользователя:

- В списке «**Основная сборка**» выделить нужный блок.
- Нажать кнопку  .
- Блок появится в списке «Сборка пользователя». После нажатия кнопки «OK», в палитре возле блока появится значок  , означающий расположение блока в секции пользователя.

Внимание: При использовании секции пользователя необходимо на вкладке **Конфигурация** задать диапазон адресов **Диапазон памяти программ пользователя**. Также возможно загрузить файл конфигурации с выделенной секцией под пользовательские блоки.

Чтобы перенести блоки обратно в основную секцию необходимо выделить имя блока в «Disabled blocks» или в «Blocks in user library» и нажать кнопку  .

Вкладка **Параметры построения** – содержит настройки для построения библиотеки для дальнейшей загрузки в память устройства.

Флаги компилятора – флаги компилятора
Флаги компановщика – флаги компоновщика (для построения исполняемого файла)
Флаги архиватора – флаги архиватора (для построения статической библиотеки)
Предопределённые символы – предопределённые символы
Подключаемые библиотеки – подключаемые статические библиотеки
Путь для поиска заголовочных файлов – папки для поиска заголовочных файлов
Путь для поиска библиотек – папки для поиска статических библиотек
Командные файлы – командные файлы распределения памяти в микроконтроллере

Для отката к настройкам заданным после установки библиотеки необходимо нажать на кнопку «**По умолчанию**».

Внимание: Подробное описание флагов и составление командных файлов смотри в документации на используемый компилятор.

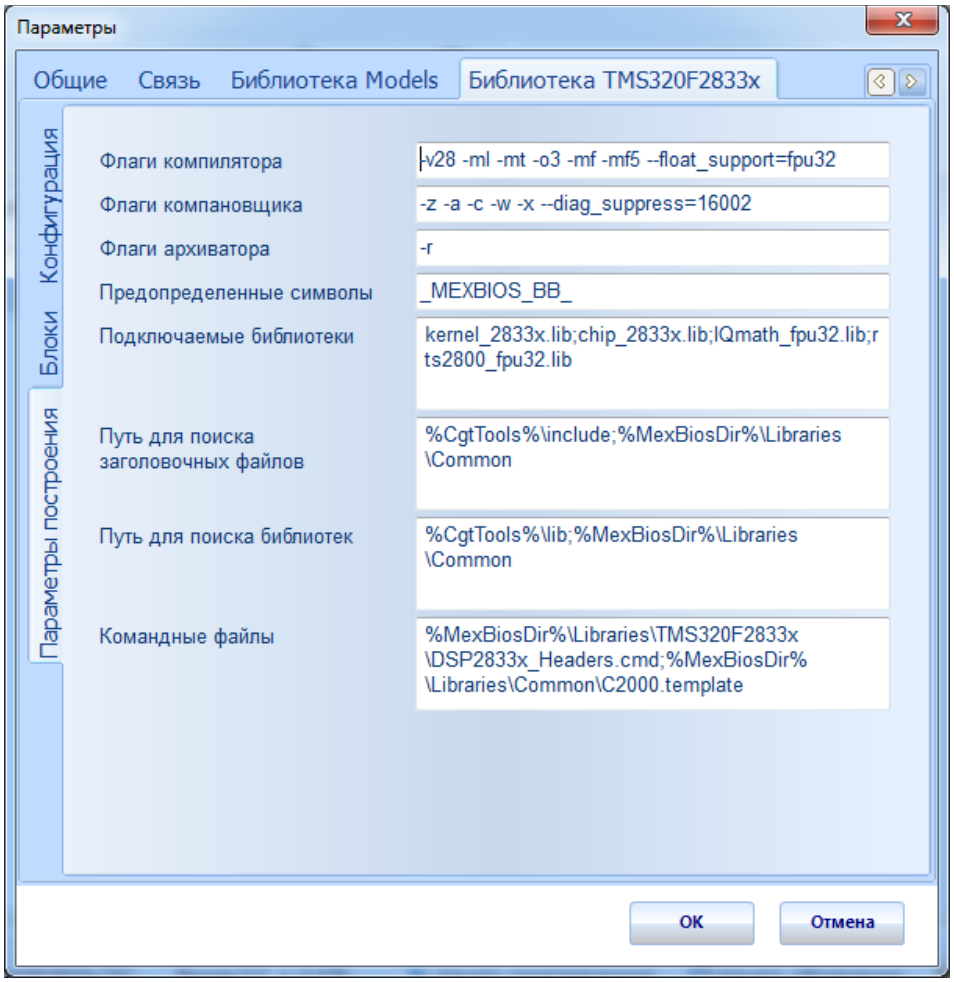


Рис. 138. Пример окна Параметры построения вкладки Библиотека

Вкладка Стартовый проект

Конфигурация стартового проекта – отражает имя файла конфигурации, содержащие все настройки для стартового проекта.

После изменения каких-либо настроек во вкладке **Стартовый проект** существует возможность сохранить их в отдельном файле конфигурации с расширением «.mbcfg». Для этого необходимо нажать на кнопку  и задать имя нового файла. При этом файлы, поставляемые при установки библиотеки, изменять не рекомендуется (они помечены атрибутом файла «Только для чтения» и недоступны для изменения).

Для загрузки ранее сохраненной конфигурации необходимо нажать на кнопку  и выбрать файл с расширением «.mbcfg». После этого в поле «Конфигурация стартового проекта» отобразится имя файла без расширения.

Папка проекта – папки нахождения фалов стартового проекта.
При выборе папки определяется переменная среды «%StartUpProjectDir%».
Имя выходного файла – название выходного исполняемого файла

Для просмотра папки хранения выходного файла необходимо нажать на кнопку  .
Папка объектных файлов – папка хранения объектных файлов

Папка для копий выходных файлов – в указанную папку будут скопированные выходные файлы библиотеки *.out и *.hex.

Флаги компилятора – флаги компилятора

Флаги компановщика – флаги компоновщика (для построения исполняемого файла)

Предопределённые символы – предустановленные символы

Подключаемые библиотеки – подключаемые статические библиотеки

Путь для поиска заголовочных файлов – папки для поиска заголовочных файлов

Путь для поиска библиотека – папки для поиска статических библиотек

Командные файлы – командные файлы распределения памяти в микроконтроллере

Исходные файлы – список исходных файлов стартового проекта.

Для отката к настройкам заданным после установки библиотеки необходимо нажать на кнопку «По умолчанию».

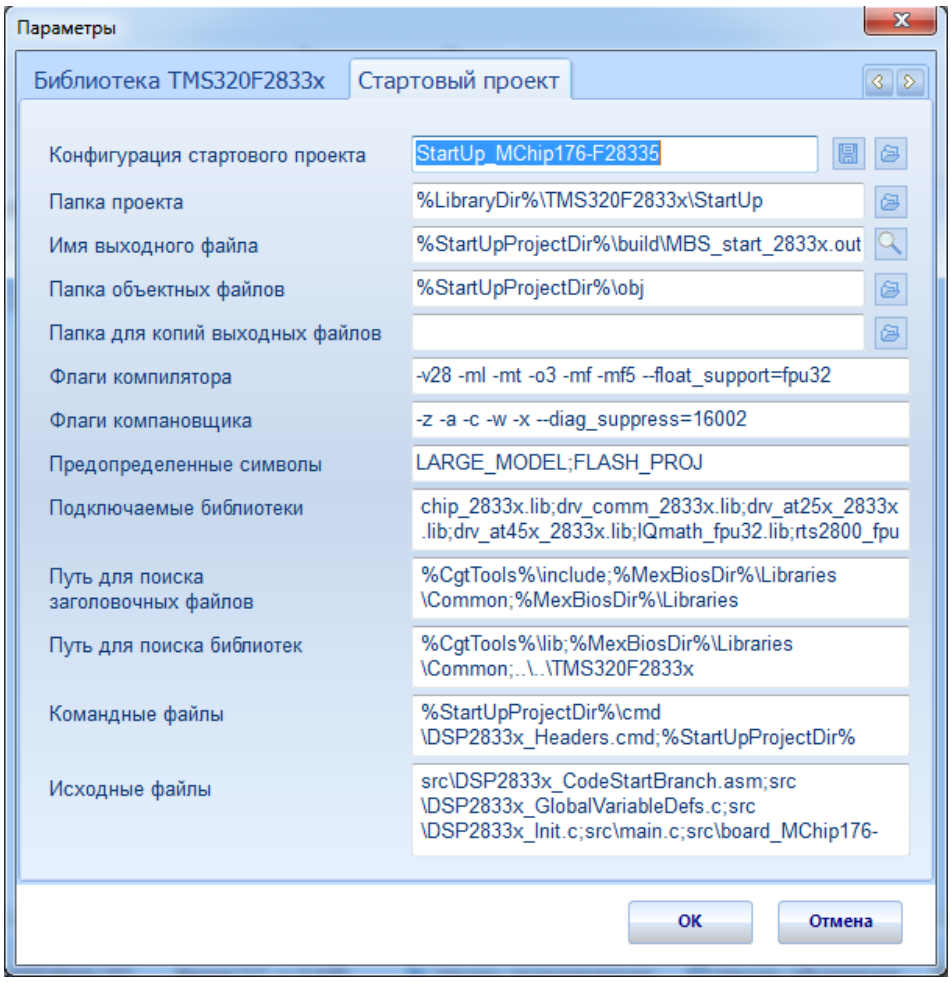


Рис. 139. Пример окна Стартовый проект

Вкладка Утилиты для построения

Папка Code Generation Tool – содержит путь к папке установленного компилятора для данного типа микроконтроллера

Командная строка компилятора – шаблон вызова компилятора через командную строку

Командная строка компановщика – шаблон вызова компоновщика через командную строку

Командная строка архиватора – шаблон вызова архиватора через командную строку

Действия после построения – действие после построения (например, создания hex-файла)

Для отката к настройкам, заданным после установки библиотеки, необходимо нажать на кнопку «По умолчанию».

Внимание: Изменение данных настроек может привести невозможности построения библиотеки и стартового проекта.

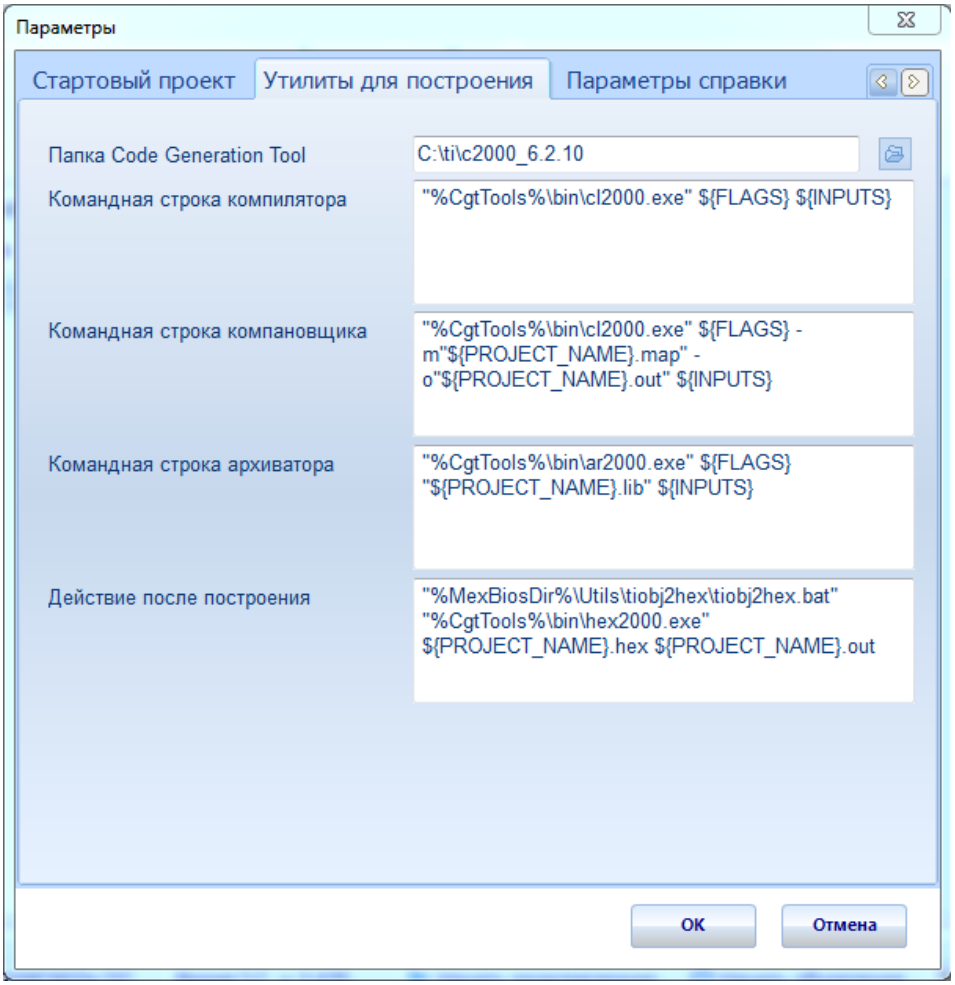


Рис. 140. Пример окна **Утилиты для построения**

Компиляция библиотеки

Для компиляции библиотеки блоков необходимо иметь установленный компилятор. Для контроллеров фирмы TI это Code Generation Tools, версии не ниже 5.2.1 (скачать нужную версию можно по ссылке http://processors.wiki.ti.com/index.php/Compiler_Releases. Для скачивания необходима регистрация на TI.com).

После компиляции блока необходимо произвести компиляцию всей библиотеки блоков, нажав кнопку **Построить** на вкладке **Правка** главного меню. Если не произвести компиляцию, то после нажатия на кнопку запуска симуляции или кнопку обмена данными с процессором будет выдано сообщение:

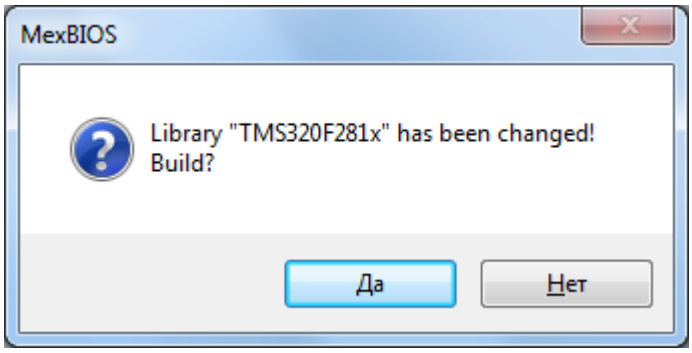


Рис. 141. Предупреждение об изменении библиотеки

Это предупреждение означает, что библиотека была изменена и необходимо произвести компиляцию. Если нажать «**Да**» запустится процесс компиляции библиотеки, если нажать «**Нет**», то компиляция будет отменена.

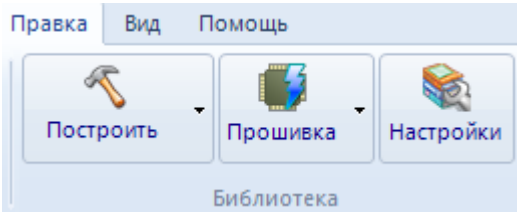


Рис. 142. Вкладка **Правка**

Кнопка «**Построить**» может находиться в двух состояниях. Если выбрано «**Построить**», то произойдёт линкование **obj** файлов всех блоков в библиотеку. Если выбрать и нажать «**Построить всё**», то произойдёт повторная компиляция всех блоков библиотеки с последующим линкованием **obj** фалов.

Библиотека блоков пользователя

Секция пользователя – отдельно выделенная память в контроллере, где пользователь может хранить свои блоки отдельно от блоков библиотеки. Для секции пользователя генерируется отдельный out файл, который необходимо прошивать в контроллер отдельно.

Для формирования библиотеки пользователя необходимо создать собственные блоки, которые далее будут перенесены в секцию памяти для блоков пользователя.

Настройка секции памяти для блоков пользователя:

1. Необходимо нажать кнопку **Настройки** на вкладке **Правка** главного меню.
2. На вкладке **Библиотека**, открыть окно **Блоки.**
3. Перенести нужные блоки в секцию пользователя **Сборка пользователя**.

Для переноса блока в секцию пользователя:

- В окне **Основная сборка** выделить нужный блок.
- Нажать кнопку  .
- Блок появится в колонке **Сборка пользователя**.

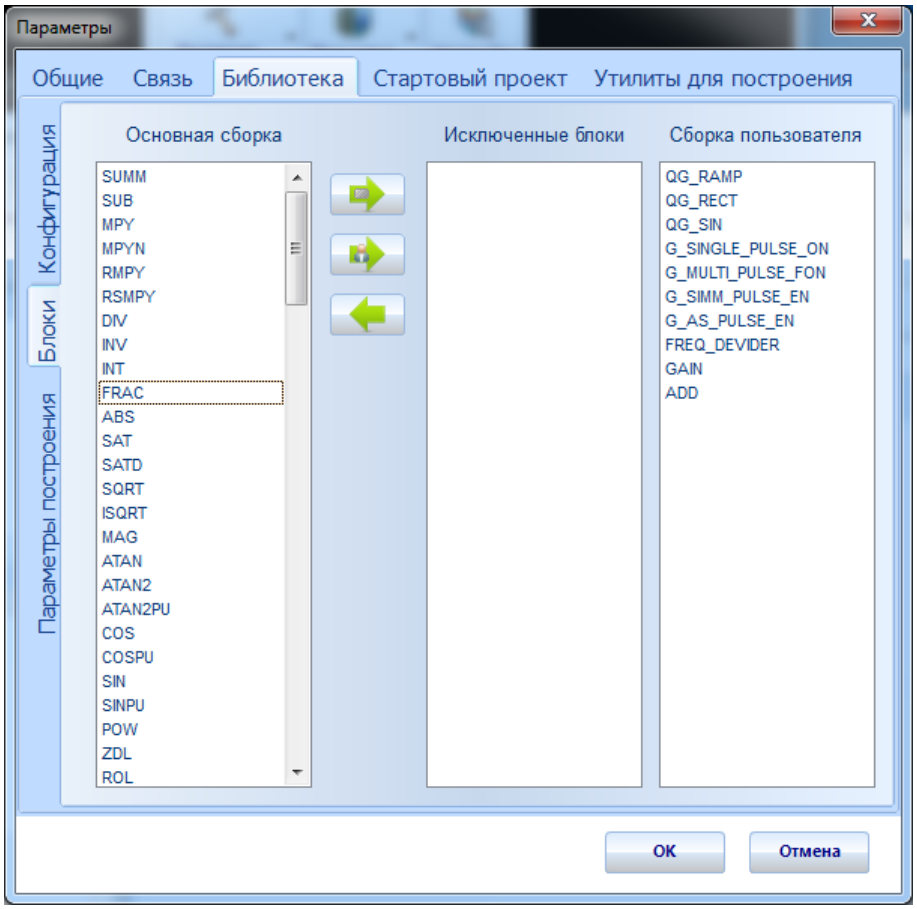


Рис. 143. Перенос блока пользователя в User section

4. Перейти на вкладку **Конфигурация**. Нажать кнопку **Загрузить сохранённую конфигурацию**.
5. Выбрать файл **Library_2812_EXT.mbcfg (**зависит от типа используемого процессора**)** в папке **Config**. В данном файле под стартовый проект выделены сектора A, B, C. Под секцию блоков

пользователя выделено два сектора D,E. Сектора F, G, H, I, J выделены под библиотеку блоков.
6. Нажать **Ok**.
7. На вкладке **Правка** главного меню, нажать кнопку **Построить**.

Необходимо перекомпилировать стартовый проект с новым **cmd** файлом настроек распределения памяти.

Чтобы перенести блоки обратно в **Основную сборку** необходимо выделить имя блока/блоков в **Сборке пользователя** и нажать кнопку  . Далее скомпилировать и загрузить библиотеку в контроллер.

Загрузка библиотеки и стартового проекта в память контроллера

Загрузку библиотеки и стартового проекта можно осуществить с помощью JTAG эмулятора и специализированной программы, предоставляемой производителем микропроцессора. Другой способ загрузки – использование возможности загрузки через последовательный интерфейс с помощью программы C2Prog (сайт http://www.codeskin.com/programmer).

C2Prog инструмент программирования flash памяти для микроконтроллеров фирмы TI серии C2000™ и MSP430™. Зачастую у пользователя нет JTAG программатора или даже нет JTAG интерфейса на устройстве. C2Prog позволяет загружать ПО микроконтроллера через RS-232, RS-485, TCP/IP, USB и CAN. Использование C2Prog подходит при работе не в лабораторных условиях, когда JTAG порт обычно недоступен.

Инструкция по загрузке с помощью C2Prog по SCI

Для загрузки стартового проекта и библиотеки блоков в память МК с помощью **C2Prog** необходимо выполнить следующие действия:

1. Перейти на вкладку **Правка** главного меню.
2. Нажать на маленький треугольник кнопки **Прошивка**.

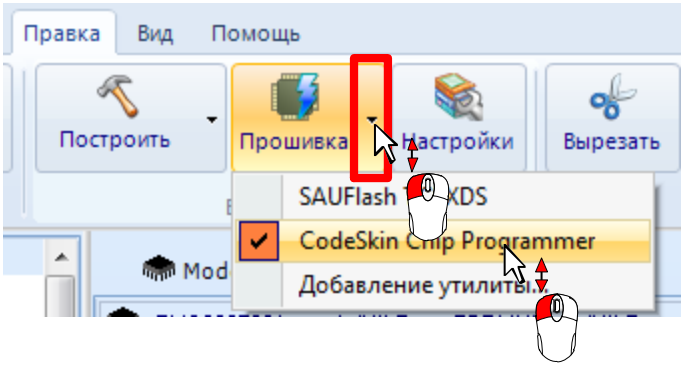


Рис. 144. Запуск **C2Prog**

3. Нажать на пункт **CodeSkin Chip Programmer**.
4. В появившемся окне нажать кнопку **Выбрать**. Окно настроек закроется.
5. Нажать на кнопку **Прошивка**.
6. Появится окно **C2Prog.** Вид окна при первом запуске представлен на следующем рисунке:

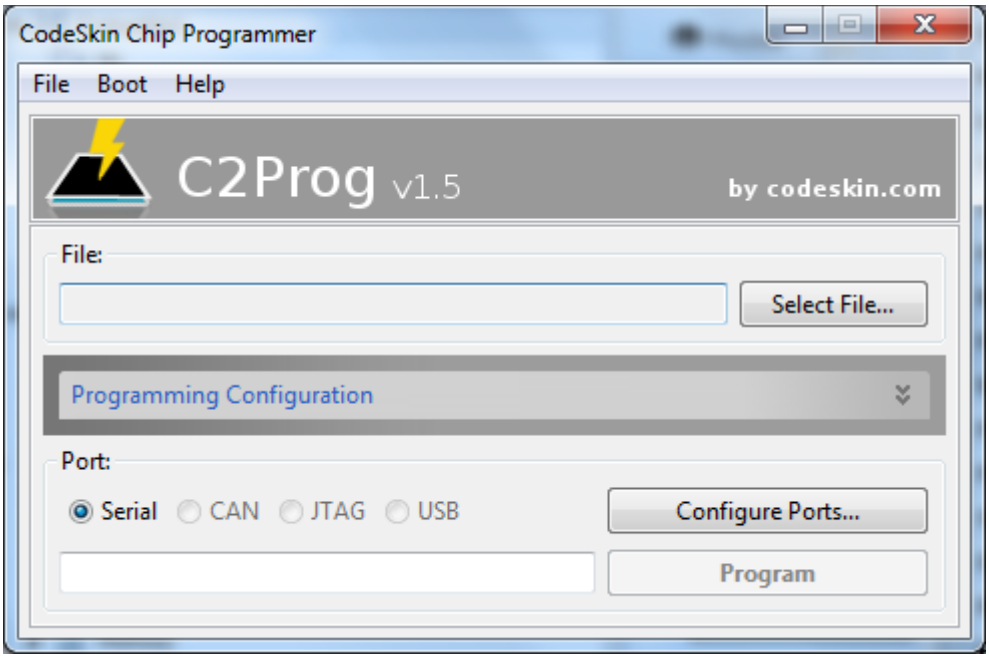


Рис. 145. Внешний вид **C2Prog** при первом запуске

7. Нажать на кнопку **Select File…** В появившемся диалоговом окне выбрать сгенерированный **.hex** файл. Нажать **Open**.

Примечание. Для библиотеки процессоров серии **281x** загружаемый **.hex** файл стартового проекта расположен по адресу:

…\MexBIOS Development Studio\Extend\TMS320F281x\StartUp\build

8. После выбора **.hex** файла стартового проекта главное окно **C2Prog** изменится:

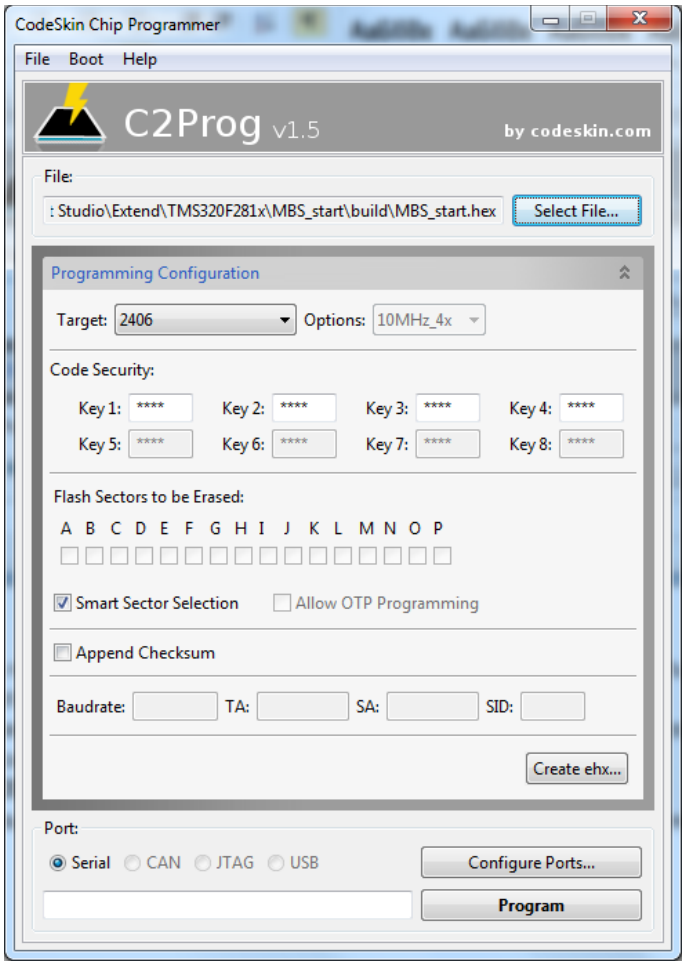


Рис. 146. Промежуточный вид окна C2Prog

9. Далее необходимо в списке **Target** выбрать нужный процессор. Покажем на примере **2812**.

10. После выбора процессора в списке **Options** необходимо выбрать тактовую частоту используемого кварца (можно посмотреть на корпусе кварца).

11. Отключенную от питания плату перевести в режим работы **SCI**. Для платы **mZdsp 2812** положение перемычек см. следующую таблицу:

Таблица 1

Положения перемычек XK3…XK6				
XK6 BOOT3	**XK5 BOOT2**	**XK4 BOOT1**	**XK3 BOOT0**	**Mode**
1	x	x	x	FLASH
0	0	x	x	SPI

Перемычка XK8 должна быть в положении **Int.**

12. Подключить плату к компьютеру.

13. Нажать кнопку **Configure Ports…**

14. В появившемся окне нажать кнопку **Scan Porst**.

15. Выбрать порт и нажать **OK**:

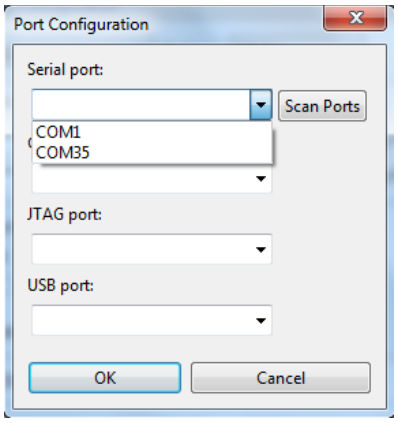


Рис. 147. Выбор порта

16. Нажать кнопку **Program**.

17. Начнётся процесс загрузки.

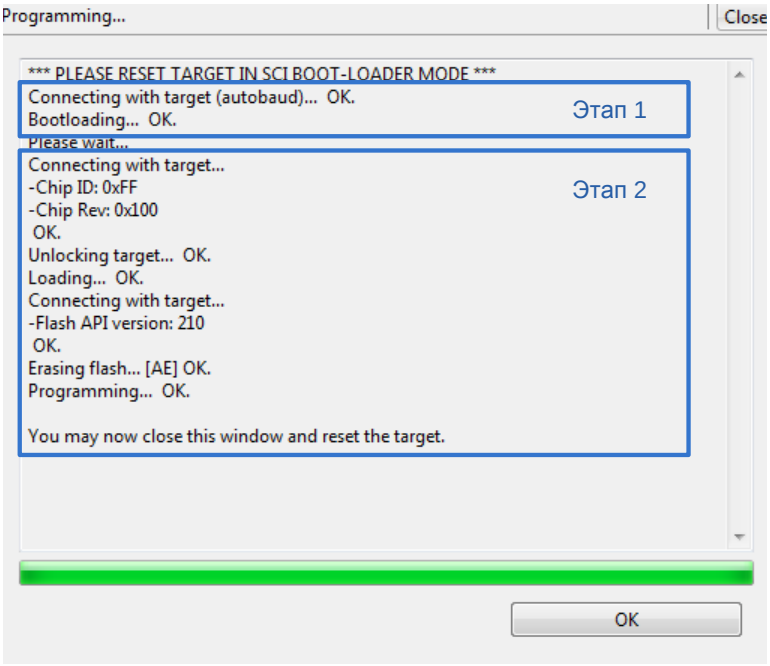


Рис. 148. Успешная загрузка в память микроконтроллера

Загрузка состоит из двух этапов. Первый этап – загрузка вторичного загрузчика. Второй этап выполнение загрузчика, очистка секторов flash памяти, загрузка программы.

18. После появления текста **«You may now close this window and reset the target»**, нажать OK, отключить питание у платы.

19. Подключить плату к компьютеру снова.

20. Выбрать **.hex** файл библиотеки блоков в папке:

…\MexBIOS Development Studio\Extend\TMS320F281x

Примечание: Файл генерируется при компиляции библиотеки.

21. Нажать кнопку **Program**.

22. После появления текста **«You may now close this window and reset the target»**, нажать **OK**, отключить питание у платы.

23. Переставить перемычки в положение **FLASH** (см. таблицу 1).

Устройство готово к использованию.

Откройте окно **MexBIOS Development Studio**.

24. Должен быть открыт файл библиотеки процессора, для которого была произведена загрузка стартового проекта и библиотеки блоков.

25. Перейдите на вкладку **Device**.

26. Настройте подключение (кнопка **Settings**).

27. Нажать кнопку **Connect**. Если всё сделано правильно – установится связь с процессором.

Функция верификации библиотеки

Для проверки идентичности блоков библиотеки зашитой в контроллер с блоками библиотеки на компьютере необходимо воспользоваться функцией **Проверить блоки** (главное меню, вкладка **Устройство**). Блоки считаются идентичными, если у них совпадают номера **GUID**.

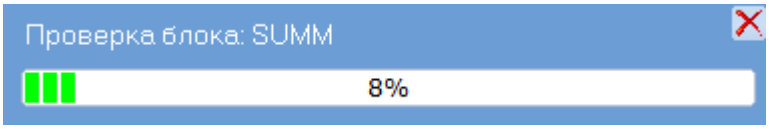


Рис. 149. Прогресс верификации библиотеки

Примечание: Функция **Проверить блоки** не отобразит блоки, которых нет в библиотеке на компьютере, но есть в библиотеке в контроллере, если все другие проверенные блоки совпадают с блоками в библиотеке на компьютере.

Если какие либо блоки из библиотек не совпадают (отсутствуют, или не одинаковый **GUID**), то после выполнения верификации появится сообщение с перечнем блоков, которых нет в библиотеке на компьютере.

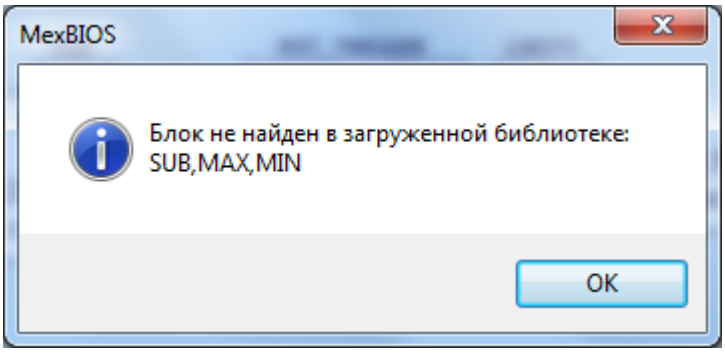


Рис. 150. Сообщение об отсутствии блоков в библиотеке

Блоки, которых нет в библиотеке, загруженной в память микроконтроллера, будут заблокированы для использования.

Рис. 151. Заблокированные для использования в проекте блоки

Для разблокирования блоков необходимо открыть **Настройки** на вкладке **Правка**. В окне **Библиотека** перейти на вкладку **Блоки**. С помощью кнопок перенести блоки из списка **Исключенные блоки** в список **Основная сборка**.

Функция Генерация кода

В данном разделе описана функция генерации кода в **MexBIOS™ Development Studio**, на языке С, по созданной схеме. Сгенерированный код выполняется процессором без участия ядра **MexBIOS**.

Сгенерировать можно только полностью программу, собранную средствами **MexBIOS™ Development Studio** в рамках одной библиотеки.

В демонстрационной версии на генерацию кода имеется ограничение в **16 384** символа. Если размер программа при генерации кода будет больше этого значения, то **MexBIOS™ Development Studio** выдаст сообщение:

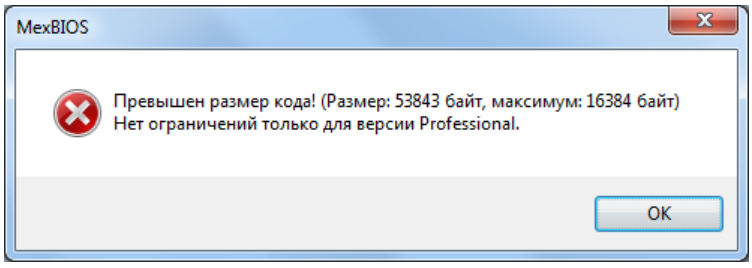


Рис. 152. Сообщение о достигнутом ограничении

Для генерации кода необходимо:

1. Открыть проект, с которого будет производиться генерация кода.
2. Открыть **Менеджер** проекта.
3. Нажать ПКМ по дереву проекта, с которого будет производиться генерация кода:

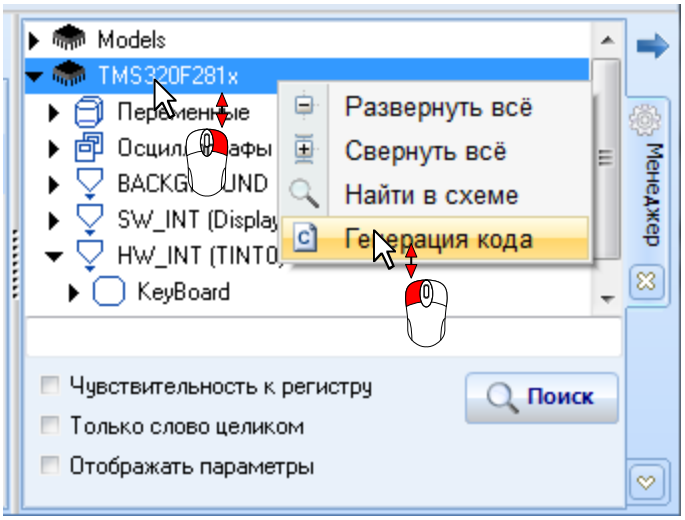


Рис. 153. Вызов функции генерации кода

4. В появившемся меню выбрать пункт **Генерация кода**.
5. Появится диалог сохранения файла.
6. Выбрать папку назначения и дать имя выходному С-файлу.
7. Если превышен лимит объёма генерации кода, то появится сообщение как на рис. 152.

Далее полученный С-файл необходимо подключить в свой проект. Покажем на примере проекта в Code Composer Studio для процессора TMS320F2812.

1. Открыть свой либо проект предоставляемого примера в **Code Composer Studio**.
2. В дереве проекта нажать ПКМ по папке **Source**.
3. В появившемся меню выбрать пункт **Add files to Project**.

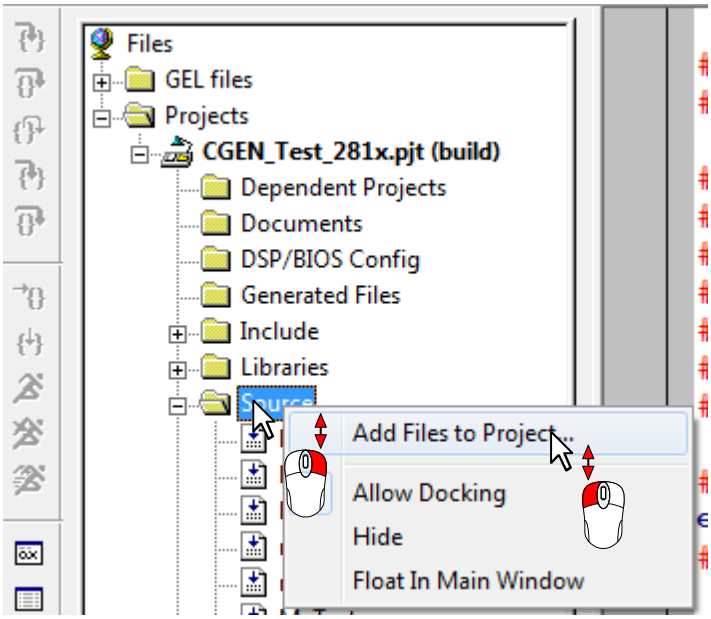


Рис. 154. Добавление сгенерированного файла в проект **Code Composer Studio**

4. В появившемся диалоге выбора файла – выбрать сгенерированный С-файл.
5. Файл появится в папке Source дерева проекта.
6. В main.c (папка Source) файле проекта объявить прототипы следующих функций:

```
// Function prototypes
void MBS_DATA_Init(void);
void MBS_BACKGROUND_Exec(void);
void MBS_TINT0_Exec(void);
```

В функции **MBS_DATA_Init** будет проходить инициализация блоков схемы. В этой функции будут выполняться все функции Init каждого блока схемы.

В функции **MBS_BACKGROUND_Exec** будет выполнять часть программы, которая в проекте mbs была подключена к событию **BACKGROUND**, а также выполнялась на программном прерывании. Если в вашем проекте не используется **BACKGROUND**, то эту функцию можно не включать в проект.

Функция **MBS_TINT0_Exec** выполняет часть программы, подключенную в проекте к аппаратному прерыванию **TINT0**. Если в вашем проекте не используется **TINT0**, то эту функцию можно не включать в проект. Также, если вы используете другой вектор прерывания (не TINT0), то имя функции **MBS_TINT0_Exec** будет другим.

7. В main функции вызвать функции инициализации – **MBS_DATA_Init** и фоновых задач **MBS_BACKGROUND_Exec**:

```
void main(void)
```

```
{
…
…

      // MBS code init
      MBS_DATA_Init();
…
      // Loop forewer
      while(1)
      {
            // MBS background task
            MBS_BACKGROUND_Exec();
      }
      }
```

8. В периодическом прерывании, выполняемом с известной частотой, необходимо вызвать функцию MBS_TINT0_Exec.

```
interrupt void CpuTimer0IsrHandler(void)
{
…
      // MBS interrupt task
      MBS_TINT0_Exec();
…
…
}
```

9. Провести компиляцию проекта.
10. Загрузить проект с помощью On-Chip Flash Programmer (меню **Tools → F28xx OnChip Flash Programmer**). Загрузка производится во все сектора.

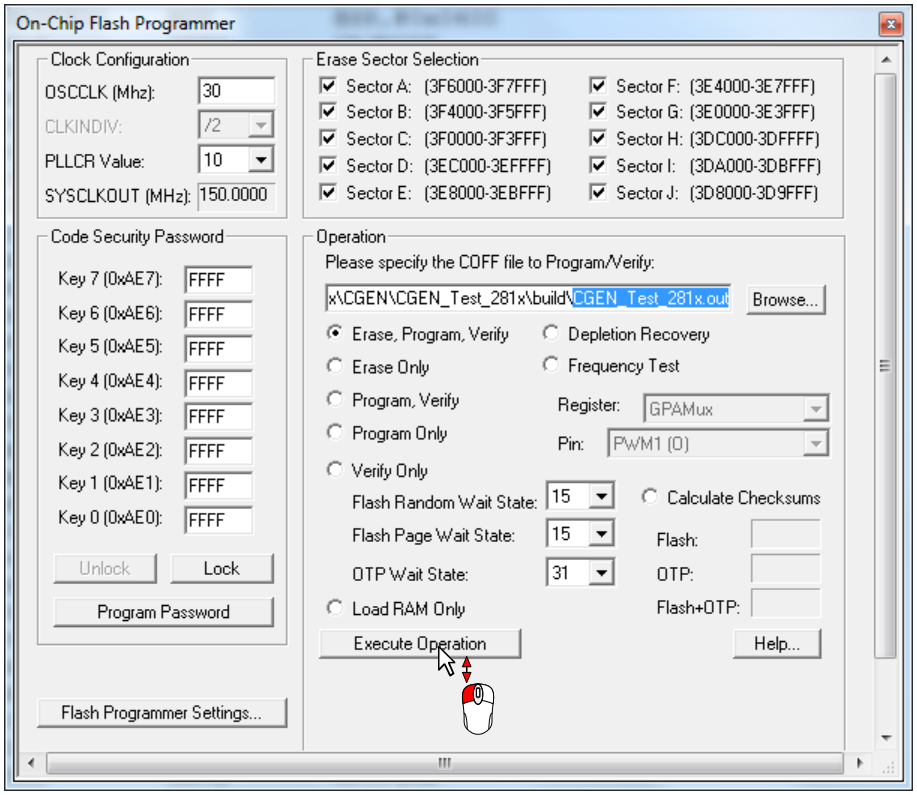


Рис. 155

11. Запустить проект на исполнение: меню→**Debug→Run**.

Генерация пользовательского интерфейса

В процессе работы в MexBIOS Development Studio пользователю доступно создание интерфейса управления с помощью визуальных компонент. Но применение MexBIOS Development Studio с завершенным устройством не всегда удобно.

Для удобства пользователя к MexBIOS Development Studio выпускается утилита GUI Player. Пользователь может на основе своего интерфейса управления сгенерировать файл *.mbe, который повторяет только интерфейс пользователя, реализованный в MexBIOS Development Studio, и способен управлять программой пользователя в контроллере по используемому пользователем интерфейсу связи.

Необходимо установить программу GUI Player.

Открыть вкладки, в которых находится органы управления, которые необходимо поместить в сгенерированный GUI.

На вкладке **Устройство** нажать кнопку **Создать GUI приложение**:

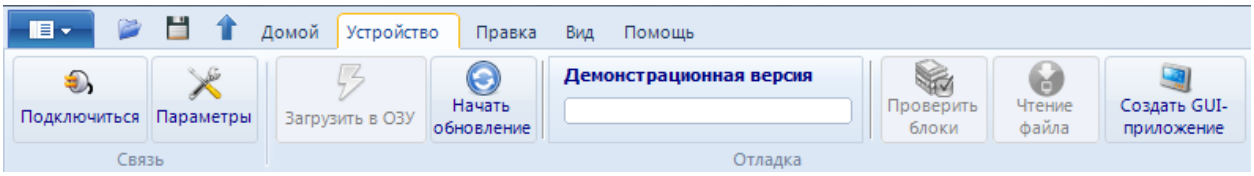


Рис. 156

Появится окно настроек:

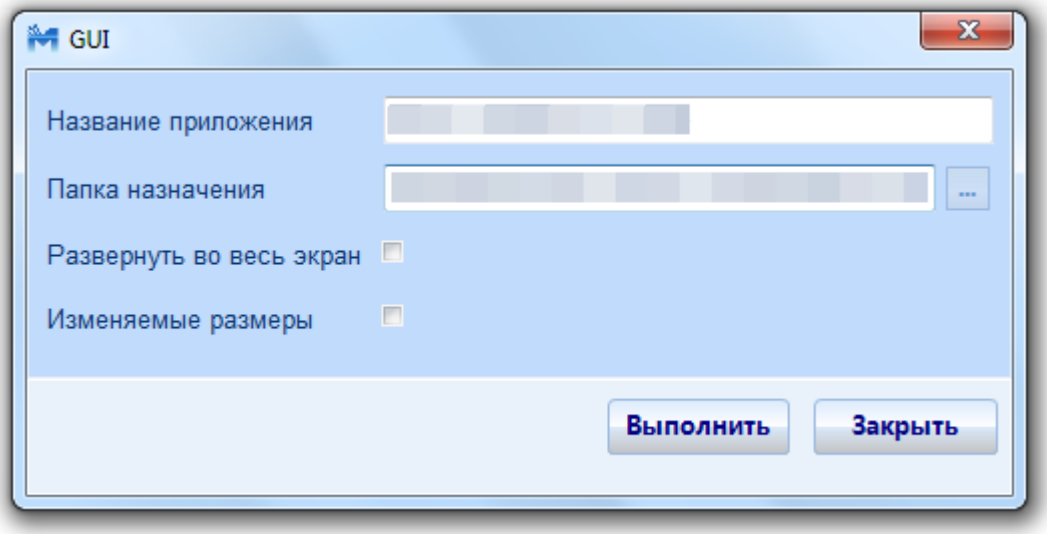


Рис. 157. Окно настроек GUI

Название приложения – название файла приложения.

Папка назначения – выбор папки, в которую будет сгенерировано приложение, причем в папку будут также скопированы картинки, используемые для оформления GUI и картинки, назначенные на кнопки управления.

Развернуть во весь экран – если пункт отмечен, то приложение будет сгенерировано по размером экрана, если пункт не отмечен, то размеры GUI будут зависеть от расположения крайних элементов

на схеме.
Изменяемые размеры – если пункт не отмечен, то GUI-приложение будет иметь статичные размеры, определяемые размером схемы, если отмечен, то размеры можно будет изменить.

Рис. 158. Служебные кнопки

В сгенерированном GUI приложении кнопка Connect/Disconnect – выводит на связь или отключает от устройства.
Start/Storp refreshing – включить остановить обновление.
Save variables to NVRAM – сохранить изменения в NVRAM.

Рис. 159. Отображение состояния и настроек связи

Пример программы

Рассмотрим пример создания визуальной программы управления светофором. Принимаем наиболее простой случай – три цвета: красный, желтый, зелёный. Переход между цветами происходит через равный промежуток времени. Реализацию осуществим сначала с помощью блоков **IF**, затем используем блок **STATE_FLOW**.

Алгоритм работы светофора:

- Горит Красный
- Горит Красный и Желтый
- Горит Зеленый
- Зеленый начинает мигать
- Загорается Желтый
- Загорается красный.

Визуализация.
Далее приведены инструкции по созданию визуального отображения светофора:
1. Создать проект **TMS320F2833x**.
2. Перейти на вкладку **TMS320F2833x**. Данная вкладка является главным полем набора программы.
3. Два раза кликнуть ЛКМ на блок **Main**. Откроется второстепенное поле набора, палитра изменится. Внутри поля набора Main создадим визуальное отображение светофора.
4. Из **Палитра→Панель элементов→Ввод данных\Вывод данных** вынести на поле набора три блока **Button** и три блока **Bulb.**
   4.1. Для всех добавленных блоков Button назначить параметр Button Group = 1. Назначение параметров смотрите главу «Визуальные органы управления».
   4.2. Для блоков **Bulb** необходимо изменить параметр **Цвет 2**. Для этого, в окне **Свойства**,

нажать на параметр **Цвет 2** и выбрать нужный цвет для каждого из трёх экземпляров **Bulb**.
5. Переименовать блоки **Bulb** и **Button**. Для этого необходимо изменить свойство **Имя** в инспекторе.
6. Соединить блоки **Button** и **Bulb**, полученная схема представлена на следующем рисунке:

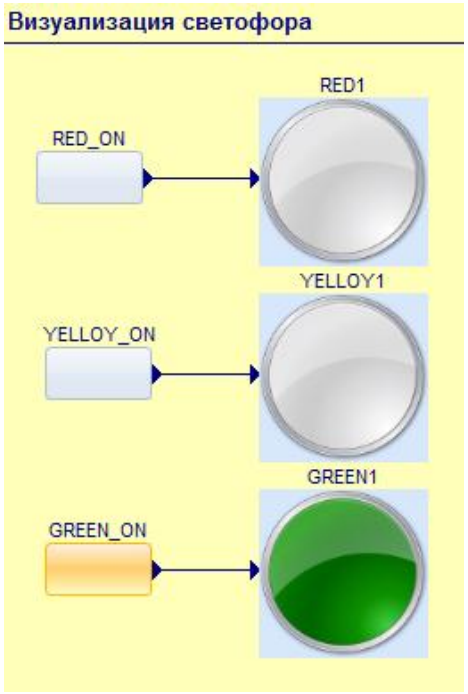


Рис. 160. Визуализация светофора

7. **Запустить процесс моделирования**.
8. Нажимая поочерёдно на кнопки **Button** убедиться, что визуализация работает.

Полученная схема показывает работу визуальных компонентов. Далее создадим систему автоматического переключения цвета на блоках **IF**. Для этого необходимо три режима работы, плюс один режим для сброса в стартовый режим. Отдельно созданный счётчик, сброс которого будет происходить по программному прерыванию.

9. Вынести на главное поле набора 4 блоков **VAR**. Переменные **GREEN**, **YELLOW**, **RED** будут включать соответствующий цвет блоков **Bulb**, созданных на предыдущем этапе. Переменная **Mode** – режим работы определённого цвета, параметр Value задать равным 1.

Рис. 161

10. Добавить на поле набора блоки **IF**, **EVENT**, **FORMULA**. Собрать следующую схему:

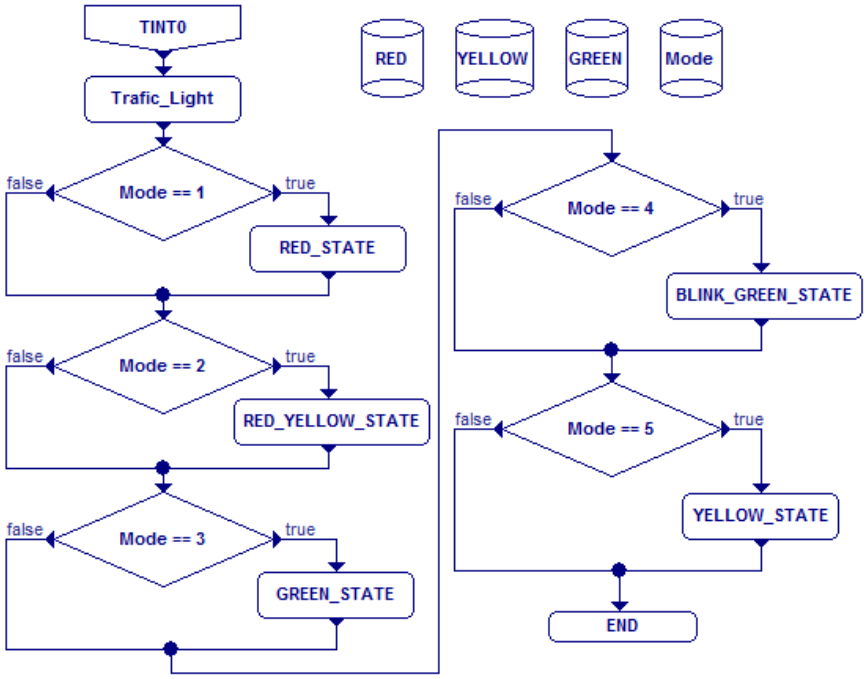


Рис. 162. Схема алгоритма работы светофора

11. Назначить условия срабатывания блоков **IF**, **EVENT**. Для наглядности переименовать блоки **FORMULA**.
12. В блоке **Trafi_Light** необходимо собрать следующую схему:

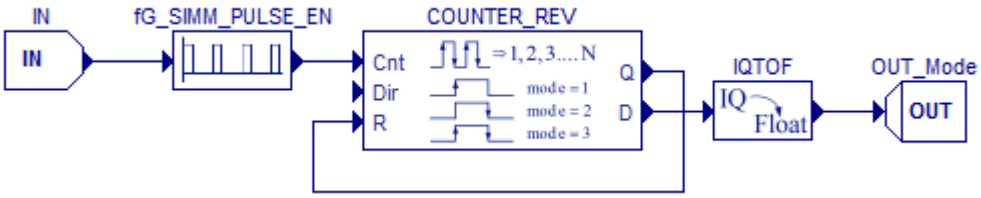


Рис. 163. Схема переключения режимов

Схема на рис. 163 представляет собой счётчик прямоугольных импульсов. Время переключения задаётся генератором прямоугольных импульсов G_SIMM_PULSE_EN, начальное значение и значение после которого происходит возвращение в первое состояние задаётся блоком счётчика COUNTER_REV.

Для задания режима необходимо OUT_Mode привязать к переменной **Mode**.

13. Формулы **RED_STATE**, **RED_YELLOW_STATE**, **GREEN_STATE, YELLOW_STATE**, одинаковы по содержанию, разница лишь в параметрах задающих блоков **IN**. В формуле **RED_STATE** единица должна подаваться на блок **OUT** привязанный к переменной **RED**, на остальные блоки, привязанные к переменным других цветов должны подаваться нули. В других формулах (**YELLOW_STATE**, **GREEN_STATE**) соответственно должна подаваться единица на цвет режима, остальные цвета должны переключаться в положение выключено.

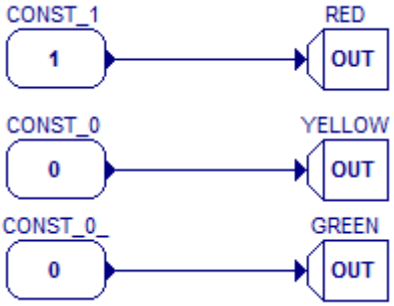


Рис. 164. Включение цвета в режиме **RED_STATE**

Формула **BLINK_GREEN_STATE** содержит генератор прямоугольных импульсов для мигания зеленым цветом.

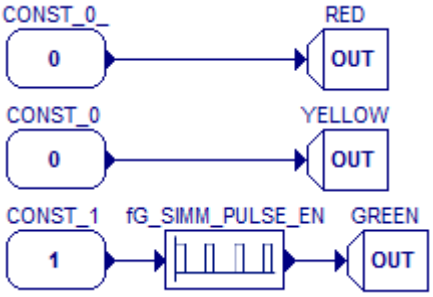


Рис. 165. Схема в блоке **BLINK_GREEN_STATE**

14. Перейти в формулу **Trafi_Light**, подключить к **Bulb** блоки **IN**, произвести привязку к соответствующим переменным, запустить процесс моделирования  и убедиться, что программа работает:

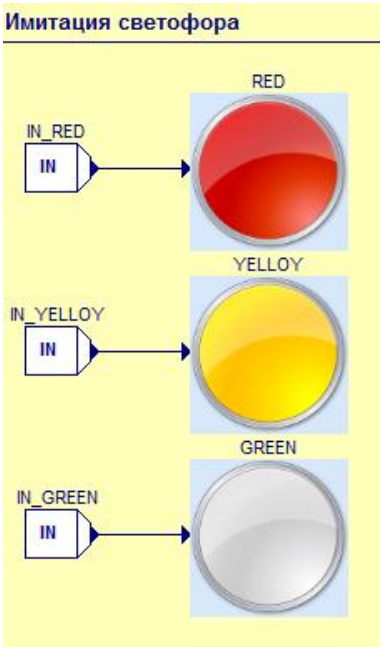


Рис. 166. Имитация светофора

Теперь покажем использование блока **STATE_FLOW**. В файле **2833x_ex_Trafic_Light_STATE_FLOW.mbp** собрана модель светофора для автотранспорта и для пешеходов. Данная модель предполагает работу как в режиме симуляции, так и в режиме эмуляции с платой **TMS320F2833x**

15. Сохраните предыдущую схему под новым именем.
16. Произведите удаление четырех блоков **IF** с ветвлениями и условием **Mode**. Оставить формулу MAIN и одно условие **IF** с формулой в ветвлении true, формулу в ветвлении переименовать в Timer to ZERO . Добавьте блок **STATE_FLOW** и измените схему. Также оставить на поле формулы **RED_STATE**, **RED_YELLOW_STATE**, **GREEN_STATE, YELLOW_STATE, BLINK_GREEN_STATE.** Полученная схема должна иметь вид:

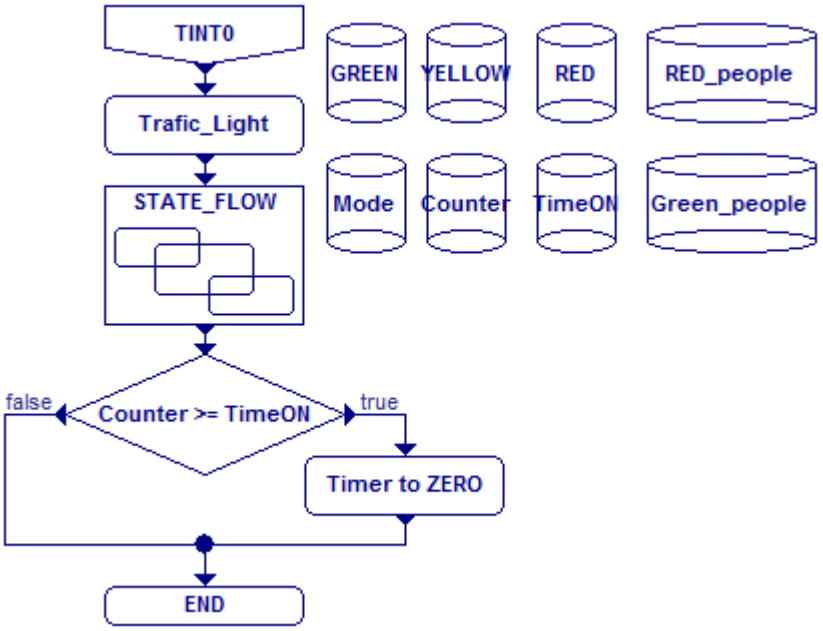


Рис. 167. Схема управления светофором с помощью блока **STATE_FLOW**

17. Добавьте четыре переменные, дайте им названия **Counter** (переменная для счётчика) и **TimeOn** (переменная для задания времени), и переменные **RED_people** и **GREEN_people (**для имитации работы светофора для пешеходов).
18. Добавьте блок **IF**. Назначьте условие Counter>=TimeOn.
19. Внутри формулы **Timer to ZERO** будет происходить обнуление счётчика. Для обнуления счётчика необходимо собрать следующую схему:

Рис. 168. Обнуление счётчика

В блок Const задать параметр Value=0. Блок OUT_Counter привязать к переменной **Counter**.

Внутри блока **Trafic_Light** удалить схему переключения режимов (рис. 162). В конечном итоге

отредактировать блок **Trafic_Light**, как показано на рис. 167 Драйвера GPI0 предназначены для индикации работы светофора на процессорной плате.

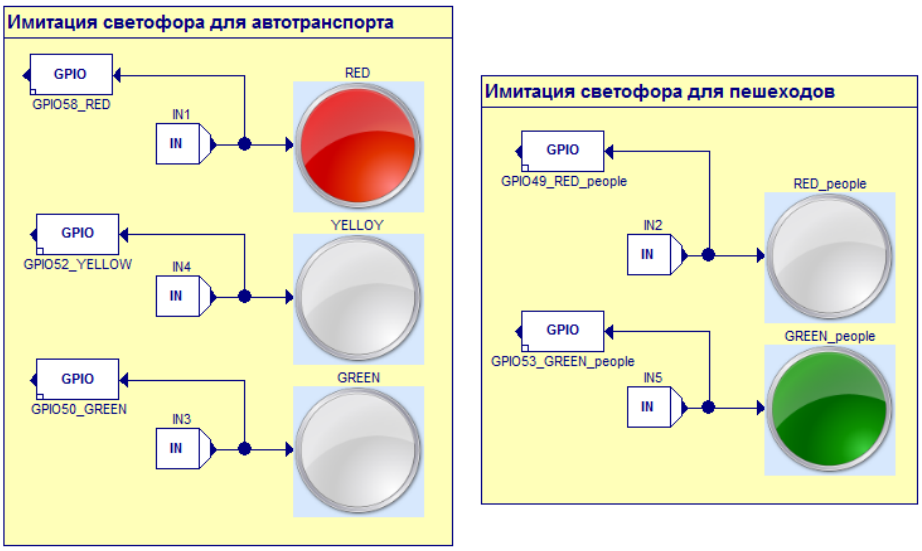


Рис. 169. Формула **Trafic_Light**

20. Зайти внутрь блока **STATE_FLOW**.
21. Добавьте блок **FORMULA**, переименовать в **COUNTER**
22. Соберите внутри блока **COUNTER** следующую схему:

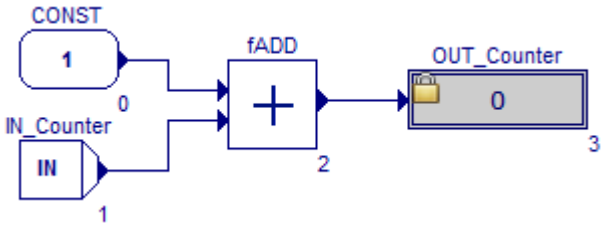


Рис. 170. Схема счётчика

Формула **Counter** содержит счетчик (рис. 170), который на каждый такт расчета процессора прибавляет с переменной **Counter** единицу и затем отправляет результат все в туже переменную **Counter.** Величина этого числа сравнивается с переменной **TimeON**, в которой уже хранится заданное число, в нашем случае это 20000. При соблюдении условий **Counter>= TimeON** состояния меняются последовательно, согласно циклу (рис. 167). Каждая формула состояний (рис.6) содержит переменные, к которым при помощи входных сигналов **IN** присваиваются значения, которые соответствуют текущему состоянию алгоритма **STATE_FLOW.**

22.1. Блок IN и OUT_Counter привяжите к переменной **Counter**.
22.2. В блок CONST задайте Value = 1.
22.3. Порядок выполнения блоков в схеме счётчика должен быть таким же, как показано на рис.162
23. Добавить пять состояний из, боковой панели элементов.
24. Соединить их и переименовать, как показано на следующем рисунке:

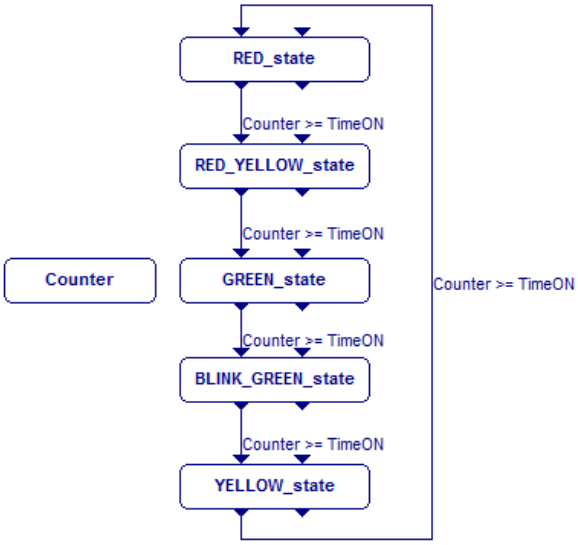


Рис. 171. Содержание блока **STATE_FLOW**

24.1. Назначить условия на линии связи. Условие Counter >=TimeOn означает, что если счётчик больше TimeOn – переключить состояние..

25. Скопировать из формул **RED_STATE**, **RED_YELLOW_STATE**, **GREEN_STATE, YELLOW_STATE, BLINK_GREEN_STATE** в соответствующие состояния.

26. В каждое из состояний добавить схемы, которые обеспечивают работу светофора для пешеходов. На следующем рисунке показаны схемы состояния **RED_state,** по аналогии дополните все состояния этими схемами**:**

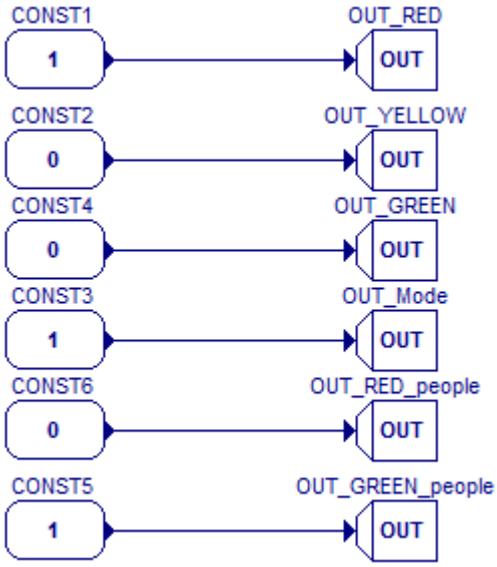


Рис. 172. Формула состояния **RED_state**

Формула **BLINK_GREEN_STATE** и **RED_YELLOY_state** содержит генератор прямоугольных импульсов для мигания зеленым цветом:

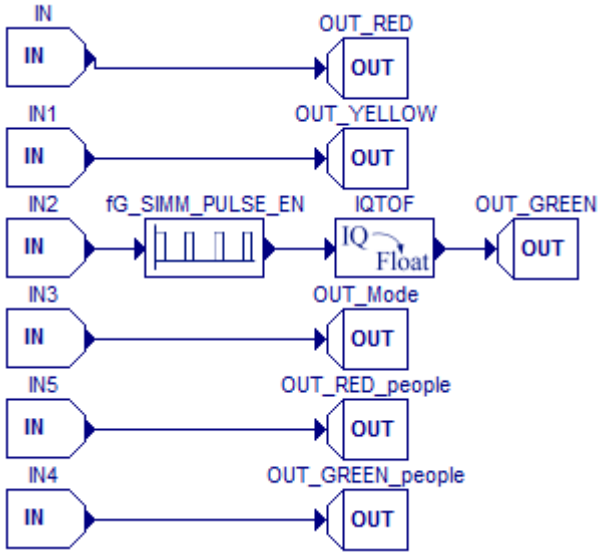


Рис. 173 Формула состояния **BLINK_GREEN_STATE**

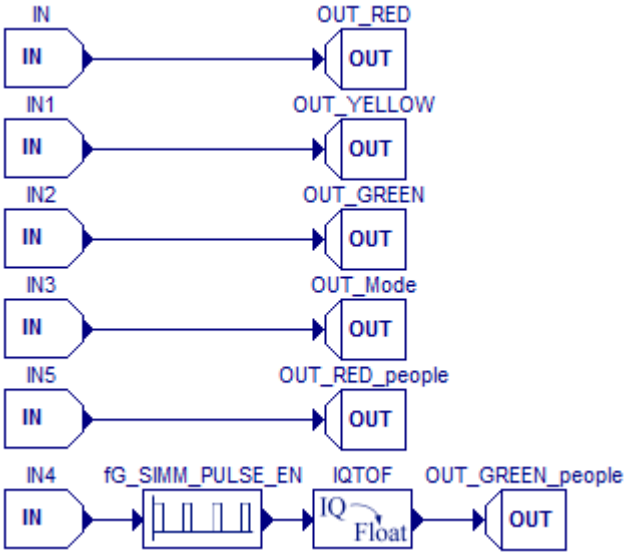


Рис. 174. Формула состояния **RED_YELLOY_state**

27. Формулы **RED_STATE**, **RED_YELLOW_STATE**, **GREEN_STATE, YELLOW_STATE, BLINK_GREEN_STATE**, расположенные на главном поле набора из предыдущей схемы можно удалить после копирования схем.
28. Перейдите в корневое поле набора. Выделите блок **STATE_FLOW**. Убедитесь, что параметр Start State установлен на **RED_STATE** (или задайте начальное состояние по своему усмотрению).
29. Перейдите в **Trafic_Light**.
30. Запустите симуляцию и убедитесь, что схема работает.
31. Для работы схемы в режиме эмуляции, подключите плату микроконтроллера **TMS320F2833x** к ПК и выполните пункт подключения **Установка связи между компьютером и чипом**.

Горячие Клавиши

Горячие клавиши редактирования	
Ctrl+X	Вырезать выделенную часть
Ctrl+C	Копировать выделенную часть
Ctrl+V	Вставить из буфера обмена
Ctrl+S	Сохранить текущий проект
Ctrl+Shift+S	Сохранить текущий проект как
Ctrl+O	Открыть проект
Ctrl+R	Вращать выделенные блоки
Ctrl+A	Выделить всё
R	Увеличить масштаб
V	Уменьшить масштаб
F	Поместить выделенный фрагмент в экран
Delete	Удалить выделенное
F11	Вызов инспектора
Ctrl+Left	Сместить выделенное влево
Ctrl+Right	Сместить выделенное вправо
Ctrl+Up	Сместить выделенное вверх
Ctrl+Down	Сместить выделенное вниз
Служебные горячие клавиши	
F1	Вызов справки по выделенному элементу
F5	Запуск моделирования
Shift+F5	Остановка моделирования
F2	Показать / Скрыть форматы сигналов
F3	Показать / Скрыть принадлежность блоков к слоям и библиотекам
F4	Добавить блок в окно **Переменные**
F11	Показать вкладку **Свойства**

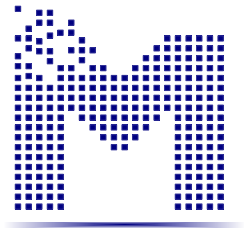
Сообщения об ошибках и предупреждениях

При возникновении ошибок следует сохранить проект. Перезапустить программу. Если ошибка будет повторяться, то обратиться в службу поддержки.

0	Неизвестная ошибка
1	Ошибка при создании проекта
3	Ошибка при создании схемы
4	Ошибка при выделении памяти для графического представления схемы
5	Ошибка конструктора связей при конструировании линии связи между блоками
6	Ошибка конструктора связей при завершении конструирования линии связи между блоками
8-15	Ошибка при нажатии мышью на различные графические объекты схемы (линии, блоки…)
16-22	Ошибка при движении мышью с захваченными графическими объектами, при движении мышью в процессе конструирования линии связи.
23-29	Ошибка при отпускании мыши над различными графическими объектами схемы (линиями, блоками…)
30-32	Ошибки при прорисовке графического представления схемы
33-36	Ошибки выделения снятия выделения с различных графических объектов
37	Общая ошибка создания блока
38	Общая ошибка создания линии
39	Общая ошибка создания узла линий
40	Общая ошибка удаления блока
41	Общая ошибка удаления линии
42	Общая ошибка удаления узла линий
43	Ошибка при поиске объекта под курсором мыши
44	Ошибка выставления свойств шрифта для графического представления схемы
45	Ошибка при вращении выделенных объектов схемы
46-63	Внутренняя ошибка графического представления при работе с линиями связей
65-73	Внутренняя ошибка графического представления при работе с узлами линиями связей
74-87	Внутренняя ошибка графического представления при работе с блоками
88-92	Внутренняя ошибка графического представления при работе с контактами блоков
94	Ошибка проекта при поиске графических объектов
95	Ошибка проекта при индексировании графических объектов
98	Ошибка сохранения проекта
99	Ошибка загрузки проекта
100	Ошибка проекта при копировании
101	Ошибка проекта при вырезании
102	Ошибка проекта при вставке
103	Ошибка проекта при записи в файл
104	Ошибка проекта при загрузке из файла
105	Ошибка при перемещении выделенных объектов
106	Ошибка при удалении выделенных объектов
107	Ошибка при определении размеров схемы
108	Ошибка схемы при поиске графических объектов
109	Ошибка при восстановлении удаленного блока
110	Ошибка при добавлении входа в схему

111	Ошибка при добавлении выхода в схему
112	Ошибка при удалении входа из схемы
113	Ошибка при удалении выхода из схемы
114	Ошибка схемы при индексировании объектов схемы
115	Ошибка схемы при сохранении
116-118	Ошибка схемы при загрузке
121	Ошибка при реорганизации истории
122	Ошибка REDO
123	Ошибка UNDO
128	Ошибка схемы при копировании
129	Ошибка схемы при вырезании
130	Ошибка схемы при вставке
131	Ошибка при удалении линии
132	Ошибка при сохранении линии
133	Ошибка при загрузке линии
134	Ошибка при сохранении линии в историю
135	Ошибка при загрузке линии из истории
136	Ошибка удаления узла линий
137	Ошибка сохранения узла линий
138	Ошибка загрузки узла линий
139	Ошибка при удалении блока
142	Ошибка при сохранении блока
143	Ошибка при загрузке блока
301-313	Ошибка вызова «callback» функции
1000	Неизвестная ошибка системы
1001-1071	Внутренние ошибки среды
10331-10334	Ошибка среды при построении матрицы
1300-1306	Ошибка инспектора параметров
1400-1420	Ошибка документа

Лицензионное соглашение



ООО «НПФ Мехатроника-Про»
Российская Федерация, 634050 г. Томск, ул. Усова 7, оф. 232
Тел:+7 (3822) 252-842
E-Mail: sales@mechatronica-pro.com
http://www.mechatronica-pro.com

Коммерческое лицензионное соглашение «НПФ Мехатроника-Про» с конечным пользователем

на продукты: MexBIOS® Development Studio, MexBIOS® Kernel, Block Builder, библиотеки процессоров для MexBIOS® Kernel

Это лицензионное соглашение ООО «НПФ Мехатроника-Про» с конечным пользователем (далее соглашение) заключено между Вами (физическим или юридическим лицом) и ООО «НПФ Мехатроника-Про», компанией, официально зарегистрированной по законам Российской Федерации, с главным офисом, расположенном по адресу Российская Федерация, 634050 г. Томск, ул. Усова 7, оф. 232 (далее «НПФ Мехатроника-Про»). Ваше использование программного обеспечения «НПФ Мехатроника-Про» означает Ваше согласие с терминами и условиями, содержащимися в данном соглашении. «НПФ Мехатроника-Про» поставляет программное обеспечение «как есть». Принимая это соглашение, Вы подтверждаете, что качество программного обеспечения удовлетворительно для Вас, и Вы не будете предъявлять требования на этом основании. Наша компания не несет никакой ответственности и не даёт никаких гарантий за прямые или косвенные убытки, включая любую упущенную прибыль, понесенных при использовании данного программного обеспечения. У Вас есть право оценить качество программного обеспечения в демо-версии. Если Вы не согласны с терминами и условиями данного соглашения, Вам не разрешается использовать программное обеспечение «НПФ Мехатроника-Про» любым образом, включая, но не ограничивая его скачивания, установки и копирования.

1. ТЕРМИНЫ

1.1. **MexBIOS® Kernel** – система выполнения задач для встроенного программного обеспечения микроконтроллеров и микропроцессоров
1.2. **Библиотеки процессоров для MexBIOS® Kernel** – набор функций, готовых к выполнению в MexBIOS® Kernel
1.3. **MexBIOS® Development Studio** – система программирования микроконтроллеров и микропроцессоров, где установлено MexBIOS® Kernel
1.4. **Block Builder** – программное обеспечение для разработки библиотек процессоров для MexBIOS® Kernel

2. ОБЩИЕ ПОЛОЖЕНИЯ

2.1. «НПФ Мехатроника-Про» является разработчиком и собственником программного обеспечения и всех включенных компонентов и его использование определяется данным соглашением. «НПФ Мехатроника-Про» предоставляет лицензию на программное обеспечение на определённые сроки, установленные на сайте «НПФ Мехатроника-Про» (www.mechatronica-pro.com).
2.2. Программное обеспечение «НПФ Мехатроника-Про» защищено законом об авторском праве и международными договорами об авторском праве, а также другими законами и договорами об интеллектуальной собственности.
2.3. Среда MexBIOS Development Studio защищена лицензионной системой, основанной на аппаратных электронных ключах HASP. HASP (*Hardware Against Software Piracy*) – это мультиплатформенная аппаратно-программная система защиты программ и данных от нелегального использования и несанкционированного распространения. Для того, чтобы среда работала в максимальном режиме, необходимо вставить электронный ключ HASP в USB-порт компьютера.

3. ОБЪЁМ ЛИЦЕНЗИИ

С учётом положений этого соглашения, «НПФ Мехатроника-Про» настоящим даёт Вам следующие права:
3.1. Демо-версия. Это соглашение даёт Вам следующие права:

Использование – Вы можете использовать каждую копию Демо-версии программного обеспечения исключительно для целей Вашей внутренней оценки, испытания и тестирования в соответствии с нижеприведёнными условиями;
3.2. Зарегистрированная версия. В случае, если программное обеспечение зарегистрировано Вами или вашим предшественником (предыдущим конечным пользователем), это соглашение даёт Вам следующие права:

Использование – Вы можете использовать каждую копию программного обеспечения для Вашего внутреннего коммерческого использования на компьютере с ключом HASP, который содержит информацию о лицензии;
Для целей настоящего соглашения “коммерческое использование” и “коммерческая деятельность” программного обеспечения означает такую деятельность, что программное обеспечение предназначено в основном для использования и получения прямого дохода лицензиатом или для того, чтобы служить основным источником, чтобы получать доход для лицензиата, включая мониторинг и испытания оборудования;
3.3. Копия для целей резервирования – Вы можете сделать одну (1) копию программного обеспечения в машиночитаемом виде только для целей резервирования. Вы должны ясно сформулировать на такой копии уведомление об авторских правах «НПФ Мехатроника-Про» и Вы не должны передавать, продавать, сдавать в аренду или давать взаймы такую резервную копию или использовать её для других целей;
3.4. Использование программного обеспечения разработчиков третей стороны для создания компонентов библиотек процессоров должно быть согласовано между Вами и этими разработчиками.
3.5. «НПФ Мехатроника-Про» не несёт никакой ответственности за поддержку приложений или за дизайн продукта потребителя. Вы ответственны за Ваши продукты

и приложения, использующие компоненты «НПФ Мехатроника-Про». Для минимизации рисков, связанных с Вашими продуктами и приложениями, Вы должны обеспечить соответствующий дизайн и меры безопасности при эксплуатации. Продукты «НПФ Мехатроника-Про» запрещены для использования в приложениях с особыми требованиями к обеспечению безопасности (как например реанимационная деятельность или искусственное поддержание жизнедеятельности), где отказ продукта «НПФ Мехатроника-Про» был бы предположительно причиной тяжёлого вреда личности.

4. ОГРАНИЧЕНИЯ

Ни в коем случае Ваши действия с программным обеспечением не должны выходить за пределы и/или противоречить пунктам 2.1 и 2.2 данного соглашения, включая, но не ограничиваясь:

4.1. Вы не можете изменять, приспосабливать или компилировать программное обеспечение, или декомпилировать, выполнять обратное проектирование (реинжениринг), дизассемблировать, или иным способом приводить программное обеспечение к виду, воспринимаемому человеком, включая, но не ограничиваясь исходным кодом;

4.2. Вы не можете делать любые модификации программного обеспечения;

4.3. Вы не можете делать любые копии программного обеспечения или документации, относящейся к нему для любых целей за исключением случаев, которые полностью соответствуют вышеупомянутым условиям относительно "Копии для целей резервирования";

4.4. Вы не можете продавать, распространять, сдавать в аренду, в лизинг, выдавать сублицензию или разрешать, чтобы программное обеспечение полностью или любая его часть было скопировано на компьютер другого пользователя;

4.5. Дополнительно к любому другому условию или положению относительно юрисдикции или любого другого аспекта международного права Вы также ответственны за исполнение законов Вашей местной юрисдикции;

5. СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПОЛИТИКА ОБНОВЛЕНИЯ

5.1. «НПФ Мехатроника-Про» будет поставлять Обновления программного обеспечения в течение стока в 1 год с даты вступления в силу этого соглашения или с даты поставки программного обеспечения для установки. Для целей настоящего соглашения “Обновление” означает модификацию программного обеспечения, которое содержит исправления ошибок и незначительные улучшения (у версии программного обеспечения большее число справа от десятичной запятой в его номере версии, чем версия программного обеспечения, которое было ранее поставлено лицензиату), и “Апгрейд” означает модификацию программного обеспечения, которая обеспечивает дополнительные признаки или выполняет дополнительные функции, не обеспеченные или не выполняемые программным обеспечением, первоначально поставленным Вам (у версии программного обеспечения новый номер версии (x.0));

5.2. «НПФ Мехатроника-Про» будет бесплатно поставлять Обновления программного обеспечения в течение срока действия лицензии, а Апгрейды потребуют дополнительной лицензии и оплаты. Настоящим устанавливается, что если Вы не покупаете у «НПФ Мехатроника-Про» дополнительную лицензию, у Вас не будет

права получать Апгрейды программного обеспечения.
5.3. По обоснованной просьбе, «НПФ Мехатроника-Про» обеспечит услуги технической поддержки и консультации по эффективному использованию программного обеспечения;

6. ПРЕКРАЩЕНИЕ СОГЛАШЕНИЯ

6.1. «НПФ Мехатроника-Про» может прекратить это соглашение в случае, если Вы нарушаете любое положение по этому соглашению. При прекращении этого соглашения Вы должны прекратить всю деятельность с программным обеспечением, удалить или уничтожить программное обеспечение, резервную копию, всю относящуюся документацию и все их копии.

7. ПРОЧЕЕ

7.1. Этот документ устанавливает полное соглашение между Вами и «НПФ Мехатроника-Про» по использованию программного обеспечения и он отменяет все прежние или настоящие договорённости или соглашения, письменные или устные, касательно этого предмета. Этот пункт может быть исключён письменным соглашением между Вами и «НПФ Мехатроника-Про». Специальные правила могут применяться к использованию определённого программного обеспечения, список которого приведён на сайте «НПФ Мехатроника-Про» и включено в это соглашение путём отсылки.
7.2. Как правило, все извещения, запросы и требования передаваемые сторонам должны быть в письменной форме и должны доставляться с уведомлением о доставке. Вместе с тем, в соответствии с этим соглашением, некоторые извещения могут быть посланы по электронной почте (e-mail).

8. ОБСТОЯТЕЛЬСТВА НЕПРЕОДОЛИМОЙ СИЛЫ (ФОРС-МАЖОР)

8.1. Обе стороны не несут ответственности за неисполнение любых обязательств по данному соглашению только в случае и в течение времени, если исполнение оных предотвращается или откладывается трудовым спором или по любой другой причине вне надлежащего контроля, таких как, но не ограничиваясь, массовые беспорядки; наводнения; война; воинственные военные действия; пожары; эмбарго; правительственные ограничения экспорта, нехватка рабочих рук, энергии, топлива, транспортных средств, или общий недостаток других предметов первой необходимости. Сторона, желающая потребовать послабление по этому пункту должна тотчас же известить другую сторону в письменном виде о возникновении и о прекращении подобного обстоятельства.

Предметный указатель