СПЕЦИАЛЬНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В МАШИНОСТРОЕНИИ

ПРОГРАММИРОВАНИЕ ПОТОЧНЫХ ЛИНИЙ

Методические указания к лабораторным работам для студентов магистратуры направления 15.04.04

САНКТ-ПЕТЕРБУРГ
2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра автоматизации технологических процессов
и производств

СПЕЦИАЛЬНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В МАШИНОСТРОЕНИИ

ПРОГРАММИРОВАНИЕ ПОТОЧНЫХ ЛИНИЙ

Методические указания к лабораторным работам
для студентов магистратуры направления 15.04.04

САНКТ-ПЕТЕРБУРГ
2020

УДК 681.9.62.503.55(073)

СПЕЦИАЛЬНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В МАШИНОСТРОЕНИИ. Программирование поточных линий: Методические указания к лабораторным работам/ Санкт-Петербургский горный университет. Сост. *А.А. Кульчицкий*. СПб, 2020. 54 с.

Тематика представленных лабораторных работ направлена на закрепление теоретических знаний и приобретение практических навыков программирования контроллеров для решения задач управления совместной работой оборудования в поточных линиях машиностроительных производств.

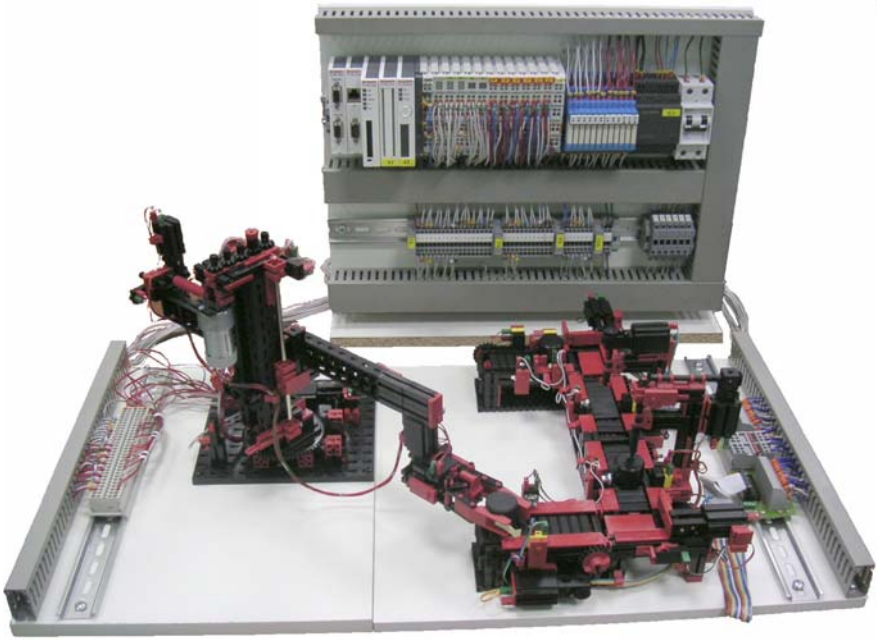
Предназначены для студентов магистратуры направления 15.04.04 «Автоматизация технологических процессов и производств».

Научный редактор проф. *В.Ю.Бажин*

Рецензент проф. *В.В. Григорьев* (Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики)

Общие указания

Изучение устройства поточной линии и основ программирования промышленных контроллеров для решения задач управления поточными линиями проводится на демонстрационном стенде «Макет поточной линии ДАСУ-02», изображение которого показано ниже.



Макет поточной линии ДАСУ-02

Лабораторные занятия дают возможность:

- закрепить на практике теоретические сведения по мобильным роботам;
- подробно ознакомиться с устройством и характеристиками мобильных роботов с различными типами движителей на примере их моделей;
- помочь овладеть практическими навыками программирования мобильных роботов;

- получить практические навыки решении задач очувствления роботов:

- выработать умение рассуждать о рабочих свойствах и степени пригодности мобильных роботов для автоматизации технологических и вспомогательных задач в горной отрасли.

Перед тем как приступить к выполнению заданий студент обязан пройти инструктаж у преподавателя. На лабораторных занятиях студенту, во избежание порчи робототехнических наборов, следует строго руководствоваться прилагаемыми методическими указаниями.

К работе в лабораториях допускаются лица, знающие инструкцию по технике безопасности, после соответствующей отметки в специальном журнале.

Порядок проведения лабораторных занятий

1. Студент должен ознакомиться с описанием соответствующей лабораторной работы и установить, в чем состоят основная цель и задачи работы.
2. Внимательно изучить теоретические положения по выполняемому заданию. Помимо сведений, приводящихся в методических указаниях, теоретическую часть, относящуюся к выполняемой лабораторной работе, необходимо освоить, опираясь на лекционный курс и литературные источники.
3. При подготовке к выполнению лабораторной работы необходимо сначала ознакомиться с устройством модели и принципом ее работы.
4. При выполнении задания по программированию необходимо сначала составить алгоритм действий, и только потом реализовывать его в программной среде.
5. Загрузить программу в контроллер и запустить на исполнение (при наличии ошибок, редактировать программу). В случае обнаружения неполадок в работе моделей транспортных роботов необходимо обращаться к преподавателю.
6. Составить отчет о результате действия программы и написать выводы по проведенной работе.

Порядок оформления отчетов по лабораторным работам

Результаты выполнения лабораторных работ обучающиеся представляют в виде отчетов.

Отчеты выполняются на листах бумаги размера А4 (верхнее и нижнее поле 2, левое 2,5, правое 1,5) Нумерация со второй страницы в правом нижнем углу. Стиль текста: шрифт набора Times New Roman; размер шрифта – 14 кегль, межстрочный интервал 1,5. На титульном листе указывается наименование предмета, специальность, шифр, фамилия, имя и отчество студента.

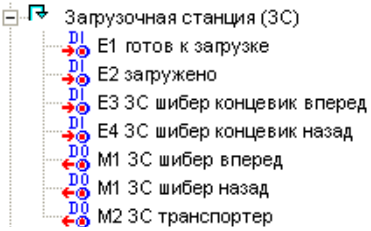
Отчет по проведению работы сдается индивидуально каждым студентом в сроки, указанные преподавателем. Результаты защиты (зачет или незачет) проставляются преподавателем в журнале учета выполнения лабораторных работ. Отчет по защищаемой работе не возвращается.

Демонстрационный стенд ДАСУ-02

Алгоритмы работы. Работа стенда заключается в слаженном взаимодействии конвейера и руки-робота. В задачу конвейера входит полная обработка детали на четырех рабочих станциях, в задачу руки же входит замыкание цикла переносом деталей с конца в конвейера вначало.

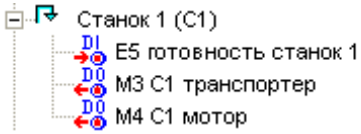
По порядку обработки детали станции конвейера называются следующим образом: загрузочная станция, станок 1, станок 2, разгрузочная станция.

Загрузочная станция принимает деталь и подает ее на обработку в станок 1. Станция оснащена двумя датчиками присутсвия детали. Первый датчик (E1 готов к загрузке) определяет, что на входе конвейера появилась деталь, и таким образом запускает работу конвейера (при любой остановке конвейера, его запуск осуществляется постановкой детали в просвет данного датчика). Второй датчик (E2 загружено) осуществляет контроль того, что деталь успешно загружена и может быть подана на станок 1.

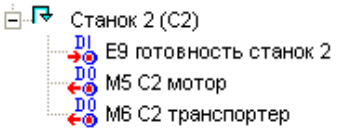


Загрузка детали осуществляется с помощью транспортера (M2 ЗС транспортер). Подача детали на станок 1 осуществляется с помощью толкателя (M1 ЗС шибер). Крайние положения толкателя контролируется двумя концевиками (E3 ЗС шибер концевик вперед и E4 ЗС шибер концевик назад). Срабатывание концевика E3 ЗС шибер концевик вперед означает, что деталь подана на станок 1 и он может приступить к ее обработке.

Станок 1 осуществляет загрузку детали, обработку и выгрузку детали на станок 2.

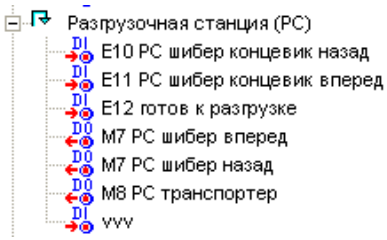


Загрузка и выгрузка детали осуществляется транспортером (М3 С1 транспортер). Датчик присутсвия детали под обрабатывающим станком (Е5 готовность станок 1) указывает на то, что загрузка детали закончена и станок должен начать обработку. Обработка производится с помощью двигателя М4 С1 мотор в течении заданного времени (данные интервал может быть изменен из интерфейса верхнего уровня). Когда обработка детали завершена транспортер М3 осуществляет выгрузку детали на станок 2.



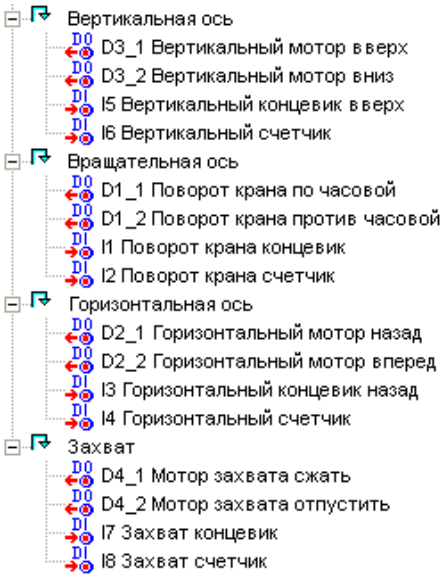
Станок 2 работает полностью аналогично станку 1.

Разгрузочная станция принимает деталь от станка 2 и перемещает ее в конец конвейера так, что рука может забрать деталь. Прием детали от станка 2 осуществляет толкатель (М7 РС шибер), срабатывание его концевика Е11 РС щибер концевик вперед дает сигнал о том, что деталь принята. Тогда транспортер (М8 РС транспортер) начинает выгрузку детали. При этом датчик присутствия (Е12 готов к разгрузке) служит промежуточной точкой, от которой отсчитывается определенное время работы транспортера, чтобы доставить деталь в конец линии, где она может быть забрана рукой (установка этого датчика в конец линии невозможна из-за недостатка свободного места для захвата руки).



Рука-робот состоит из трех осей и захвата. Каждая ось (в том числе и захват) управляется одним реверсивным двигателем, которому в программе соответствует два управляющих дискретных сигнала — соответственно для движения в одну и другую сторону. Включение двух сигналов одновременно должно быть программно запрещено, выключение обоих сигналов приводит к остановке двигателя. Ось также снабжена одним концевиком и датчиком оборо-

тов, с помощью которых может быть определено положение оси. Такая конфигурация требует инициализации крана, при которой все оси перемещаются в положения, в которых срабатывают концевики. После этого возможно определение положение оси с помощью датчика оборотов — положение оси задается в количестве оборотов. При включении контроллера программа осуществляет данную инициализацию и устанавливает все оси в положение готовности для захвата детали. Такая же инициализация проводится также после каждых 10 циклов переноса детали с целью исключения возможных накопившихся ошибок в счетчиках положения осей.



Захват детали инициируется моментом когда конвейер закончил выгрузку детали. Дальнейшие движения руки жестко заданы в координатах осей за неимением каких-либо внешних датчиков для определения положения руки. Поэтому для правильной работы стенда важно всегда соблюдать правильное расположение оснований конвейера и руки.

Назначение отдельных частей проекта ПО контроллера

Проект ПО контроллера для удобства разбит на программы, каждая из которых отвечает за определенную часть алгоритма. Основной алгоритм расположен в месте работы Таймер:

- Управление конвейером:
  - **st1** — управление загрузочной станцией
  - **st2** — управление станком 1
  - **st3** — управление станком 2
  - **st4** — управление разгрузочной станцией

- **protect** — контроль за авариями конвейера: при превышении установленных в данной программе таймаутов выдается сигнал об аварии и конвейер останавливается.

- Управление рукой:
  - **Initarm** — алгоритм инициализации руки для установки ее осень на концевые положения. На втором листе данной программы также заданы значения положений осей в характерных точках алгоритма. Данные значения затем используются в остальной части алгоритма.
  - counter — расчет текущего положения осей
  - **TkCub** — захват детали
  - **DOWN** — обработка всех режимов, при которых необходимо опускание руки вниз (перед захватом и откпусканием детали)
  - **UP** — обработка режимов, при которых необходимо поднятие руки (после захвата и отпускания детали)
  - **LeftIn** — осуществляет поворот руки налево (для отпускания детали) и втягивания стрелы для выхода на позицию опускания
  - **RightOut** — осуществляет поворот руки направо (для захвата детали) и выпускание стрелы для выхода на позицию захвата детали
  - **GiveCub** — отпускание детали их захвата
  - **ProtArm** — контроль за авариями руки: при превышении установленных в данной программе таймаутов выдается сигнал об аварии и рука останавливается.

Программа **Inpprg** места работы Входы содержит функциональные блоки, отвечающие за ввод/вывод сигналов контроллера. Программы места работы Интерфесы задают интерфейс обмена контроллера с верхним уровнем.

Лабораторная работа №1

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОНТРОЛЛЕРОВ В СРЕДЕ «ПОЛИГОН»

1.1 ЦЕЛЬ РАБОТЫ

Изучение основ программирования промышленных контроллеров на языке FBD для решения задач рефлекторного управления совместной работы оборудования в машиностроительной отрасли.

ЗАДАНИЕ

В качестве примера иллюстрирующего технологию программирования в Полигоне для контроллера Beckhoff CX1000-000 разработаем простейшую программу запуска станка первой рабочей станции (рис. 1.1) После запуска управляющей программы запустить станок на 5 секунд.

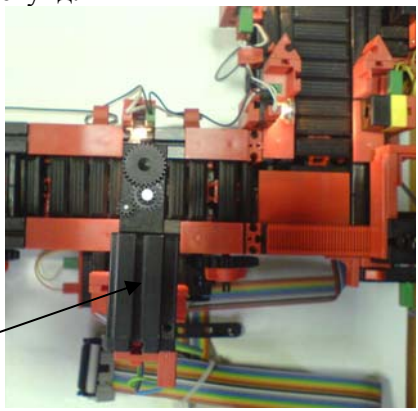


Рис. 1.1. Фрагмент поточной линии

ОБЩИЕ УКАЗАНИЯ

Последовательность действий для создания проекта содержит два основных этапа:

1. Сформировать аппаратную часть проекта, т.е. сформировать дерево проекта из библиотечных блоков в **Конфигураторе** и пре-

образовать ее в формат программной части (надо выполнить команду **Передать проект в редактор** в меню **Проекты**).

2. Создание и редактирование алгоритмов программы в **Графическом редакторе**. Для этого необходимо закрыть проект в **Конфигураторе** (во избежание ошибок при совместном доступе к одним и тем же базам данных) и запустить **Графический редактор**.

Рассмотрим подробнее эти этапы

Выбор элементов аппаратной части проекта производится из библиотечных блоков в окна **Библиотека** в первом окне **Конфигуратора** (см. рис. 1.2).

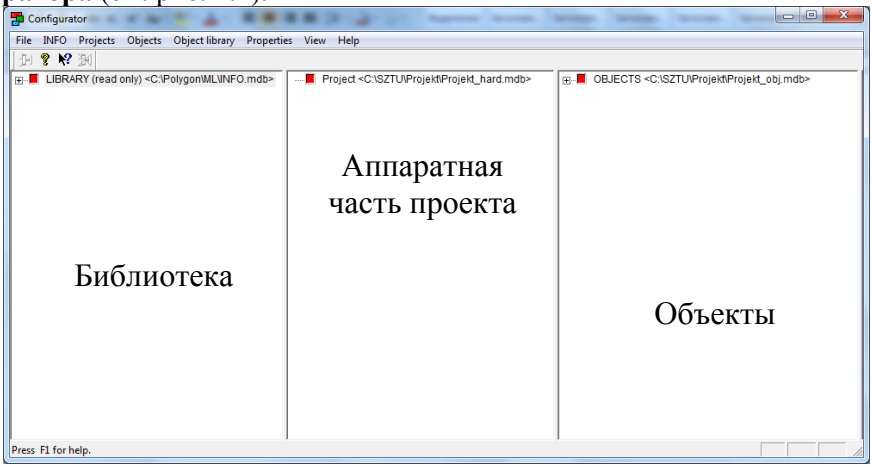


Рис. 1.2. Окно конфигуратора

Аппаратная часть проекта формируется в виде дерева во втором слева окне **Конфигуратора**. Дерево состоит из узлов, далее называемых элементарными блоками. Поскольку библиотека также представлена деревом, то формирование аппаратной части заключается в простом перетаскивании мышью элементарного блока из библиотеки в проект. (см. рис. 1.3).

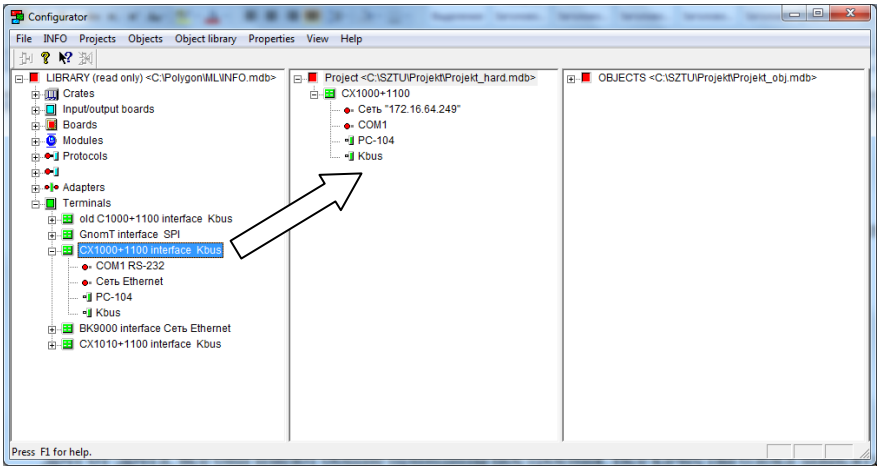


Рис. 1.3. Окно конфигуратора

Все остальные необходимые операции над элементарным блоком становятся доступными по нажатию на нем правой кнопки мыши в выпадающем контекстном меню.

Элементарный блок в Конфигураторе

Не смотря на то, что в аппаратной части проекта присутствуют блоки, сильно отличающиеся друг от друга, все они имеют общие принципы построения. Все качества блока зависят от того, к какому классу он принадлежит, а также от его свойств.

Свойства блока можно разделить на два класса:

- *информационные* – свойства, которые несут информацию о характеристиках самого блока (например, базовый адрес платы или прерывание COM порта)
- *конструкторские* – свойства, влияющие на создание дерева аппаратной части проекта, устанавливая правила согласования различных блоков; например, для слота и платы, которую необходимо вставить в этот слот, должно существовать свойство «Шина платы/слота» с одинаковым значением.

Свойства блока можно изменить следующим образом: выделите блок, с помощью нажатия правой кнопки мыши выведите контекстное меню и в нем выберите команду **Свойства**.

Конструкторские свойства сравниваются, когда блок из библиотеки добавляется к блоку в аппаратной части, и если значения этих свойств не совпадают, добавление запрещается.

Чтобы свойство блока из библиотеки имело определенное значение, надо либо ввести это значение, если возможно одно значение (на данное свойство нет списка возможных значений), либо в списке значений у требуемого значения изменить тип выбора на «по умолчанию», а у остальных – на «по заказу».

Каждый блок может иметь как те, так и другие свойства. Подробнее о свойствах будет рассказано ниже при обсуждении конкретных блоков.

Все блоки аппаратной части проекта делятся на следующие классы:

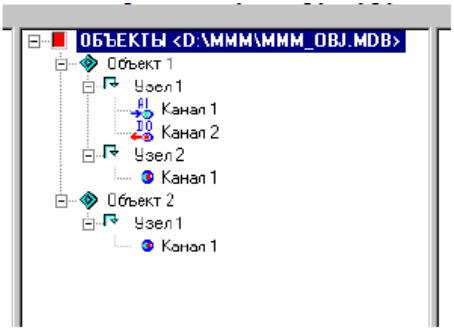
-  Крейт
-  Слот
-  Плата ввода/вывода
-  Микросхема платы
-  Неинициализированный регистр платы
-  Инициализированный регистр платы
- 

Канал платы ввода/вывода
-  Процессорные и интерфейсные платы
-  Порт интерфейса
-  Интерфейс
-  Группа интерфейса
-  Переменная группы интерфейса
-  Модуль
-  Терминал

Объекты

Объекты проекта отображаются также в виде дерева в третьем слева окне Конфигуратора. Объекты служат для непосредственной связи объекта управления с проектируемой системой управления подобно тому, как, например, датчики, установленные на объекте, проводами соединяются с входами плат ввода/вывода контроллера. Таким образом дерево объектов отображает структуру объекта

управления и в отличии от дерева аппаратной части имеет жестко определенную структуру:



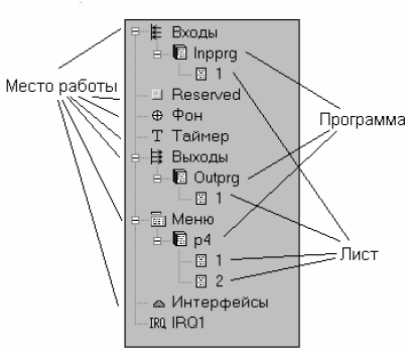
В качестве объекта может выступать, например, резервуар для слива жидкости, тогда одним из узлов может быть заслонка на входе в резервуар, а его каналами: DO открыть заслонку, DO закрыть заслонку, AI положение заслонки и т.д.

С аппаратной частью проекта объекты связываются посредством каналов (перетаскиванием канала из объектов в канал аппаратной части), т.е. установлением связей между каналами объектов и аппаратной части. Причем один канал объектов может быть связан со многими каналами аппаратной части, но не наоборот. Таким образом, для конечного проекта важны только имена каналов (они в конце концов попадут в **Графический редактор** в виде комментариев к входам/выходам), которые задаются в дереве объектов, а остальная информация существует только для информативных целей. Дерево объектов можно распечатать, для этого надо щелкнуть на нем левой кнопкой мыши и выбрать команду **Создать файл** спецификации в меню **Объекты**. При этом будет создан текстовый файл следующего вида (для примера дерева объектов представленного выше):

```
> Файл спецификаций: D:\MMM\MMM_OBJ_spec.txt
> База данных: D:\MMM\MMM_OBJ.MDB
-Объект 1
        |-Узел 1 -
                | |-<AI> Канал 1 - Комментарии канала 1
                | |-<DO> Канал 2 - Комментарии канала 2
        |-Узел 2 -
                | |-<нет> Канал 1 - Комментарии канала 1
-Объект 2
        |-Узел 1 -
                | |-<нет> Канал 1 - Комментарии канала 1
```

После подготовительных операций, проведенных в **Конфигураторе**, по созданию аппаратной части проекта, и преобразование этой аппаратной части к формату программной части, в Графическом редакторе должна быть проведена основная работа по созданию алгоритмов и отладке программного обеспечения. Поскольку Графический редактор ориентирован на работу с алгоритмами, идеология представления программы в нем является блочной. Таким образом, общий алгоритм работы программы разбит на изолированные функциональные блоки, каждый из которых выполняет определенную функцию.

Структура программной части проекта.



Проект представляет собой набор Программ находящихся в различных **Местах** работы. Для удобства просмотра, программы разбиты на **Листы**, каждый из которых является составной частью программы. Количество листов определяется пользователем и неограниченно. **Лист**, в свою очередь, содержит **Функциональные блоки**, соединенные между собой связями.

Основные функциональные части экрана Графического редактора:

1. Два **Рабочих окна**: равнозначные области для вывода на экран **Листов**, входящих в **Проект**. Два окна введены для удобства работы (например, для проведения связей между блоками, находящимися на разных листах).

2. Панель выбора текущего листа содержит две области выбора **Места работы**, **Программы** и **Листа** для каждого из **Рабочих окон**. При выборе из списка листа, он отображается в соответствующем Рабочем окне и назначается текущим (это используется в

таких операциях, как, например, создание блока).

На панели имеются также кнопки пролистывания листов (стрелки) для перехода к следующему или предыдущему по порядку листу. Под областью выбора отображаются комментарии к листам, открытым в Рабочих окнах. Комментарии можно изменить, щелкнув левой кнопкой мыши на их изображении.

В правом нижнем углу главного окна Графического редактора расположено имя текущего листа: 

Существует несколько типов мест работы:

Входы	Выполняются в таймерном прерывании
Таймер	
Фон	Выполняются в фоновом процессе
Интерфейсы	
Меню	
IRQ (Обработчик прерывания)	Выполняется по факту появления прерывания

При создании проекта в него автоматически добавляются все перечисленные типы мест работы, имеющиеся в данной версии программы (кроме обработчиков прерываний IRQ). В процессе работы невозможно добавление или удаление Места работы (за исключением IRQ). Такая структура проекта определяется в первую очередь требуемой структурой создаваемого программного обеспечения.

Программа

Программа делится на листы, на которых в свою очередь можно создавать функциональные блоки и проводить связи. В программу, не имеющую ни одного листа, нельзя добавить блок. При создании программы она наследует все свойства от Места работы, на котором была создана, и затем передает их своим листам. Таким образом, Программа является связующим звеном между Местом работы, от которого зависит, как будут выполняться функциональные блоки, и Листами, на которых эти блоки можно размещать.

Программа считается текущей, если один из ее листов является текущим.

Параметры программы:

имя – строка, уникальная для всего проекта, состоящая не более, чем из 8 символов, должна иметь формат, допустимый для имени файла (не содержать символов русского алфавита, символов ! " « » № ; % : ? * ( ) -=+);

порядок выполнения – целое положительное число, характеризующее порядок выполнения программы в проекте, порядок выполнения можно изменить после создания программы, порядок выполнения сквозной для Места работы, в котором находится программа.

Лист

Понятие **Листа** введено для удобства представления **Программы**. Таким образом, программа делится на листы, у каждого из которых есть условный номер (в порядке этих номеров выполняются блоки в окончательной программе), и могут быть комментарии. Листы можно создавать и удалять.

Графически лист представляется в виде двух областей:

1. **Рабочая область** :

В этой области располагаются принадлежащие листу функциональные блоки.

Соответственно все манипуляции с функциональными блоками (перемещение, создание, удаление, проведение связей) проводятся здесь.

2. **Поля**:

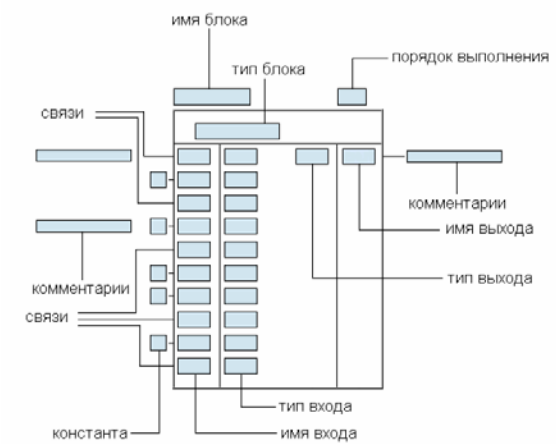
Область, содержащая информацию о входах и выходах блоков, находящихся на других листах, связанных с входами и выходами блоков данного листа. На листе существует два поля: левое, несущее информацию о входах, и правое, несущее информацию о выходах.

Вход или выход на поле обозначается следующим образом:

Для того, чтобы быстро найти вход или выход, указанный на поле, необходимо два раза щелкнуть на нем левой или правой кнопкой мыши. После этого в соответствующем **Рабочем окне** (в левом, если левая кнопка, в правом - если правая) появится лист, которому принадлежит данный вход или выход. Последний помечается красной стрелкой, которая исчезает после щелчка левой кнопкой мыши на блоке.

Функциональный блок

Функциональный блок представляет собой элементарный алгоритм.



Параметры блока:

имя блока – строка, уникальная для всего проекта не должна содержать символов русского алфавита, символов !”№;%:?*()_+=- и не начинаться с цифры. Имя можно поменять после создания блока;

порядок выполнения – положительное целое число, характеризующее порядок выполнения блока на листе (порядок выполнения блока в конечной программе определяется порядком выполнения Программы, номером Листа и его собственным порядком выполнения), для корректного выполнения программы рекомендуется не использовать одинаковые порядки выполнения для разных блоков. Порядок выполнения можно менять после создания блока;

тип блока – имя типа функционального блока (из библиотеки функциональных блоков). Тип задается при создании ФБ и не меняется в процессе работы;

имя входа – строка, уникальная для данного блока, состоящая не более чем из 3 символов, может содержать цифры. Имя входа не доступно для редактирования пользователем, а задается автоматически при создании блока;

тип входа – имя типа данных, характеризующего данный вход. Задается автоматически при создании блока и зависит от типа функционального блока. Тип входа учитывается при создании связей и изменении ***константы*** на входе;

константа – константное выражение, формат которого зависит от ***типа входа***. Константа задается на входе, не связанном ни с каким выходом другого блока и определяет в этом случае значение входа при выполнении программы;

имя выхода – строка, уникальная для данного блока, состоящая не более чем из 3 символов, может содержать цифры. Имя выхода не доступно для редактирования пользователем, а задается автоматически при создании блока;

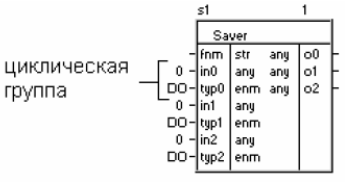
тип выхода – имя типа данных, характеризующего данный выход. Задается автоматически при создании блока и зависит от типа функционального блока. Тип выхода учитывается при создании связей.

Входы

Входы функционального блока могут быть разного типа:

1. Циклические – входы, количество которых может быть переменным. Их число задается при создании блока и может быть изменено в процессе работы (см. раздел Добавление/удаление входов/выходов).

2. Циклические группы – входы, объединенные в группы, количество которых в свою очередь может быть переменным. Число групп задается при создании блока и может быть изменено в процессе работы. Группа



аналогична структуре в Си.

3. Нециклические – входы, количество которых не может быть изменено и задается автоматически при создании блока.

Все эти типы входов в свою очередь делятся на:

1. Константный вход – вход, на котором должна быть задана константа и не может быть проведена связь.

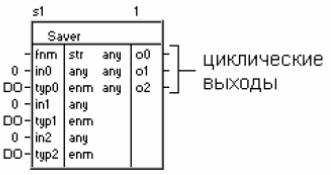
2. Обязательный вход – вход, на который должна быть проведена связь и не может быть задана константа. Если на все входы данного типа не будут проведены связи, то при запуске трансляции проекта будет выдана ошибка. Данный вход обозначается символами ??? в поле констант.

3. Необязательный вход - вход, на который может (не обязательно) быть проведена связь и не может быть задана константа. При отсутствии связи вход не используется в программе.

Данный вход обозначается символом # в поле констант.

Выходы функционального блока могут быть двух типов:

1. Циклические – выходы, количество которых может быть переменным. Их число задается при создании блока и может быть изменено в процессе работы.

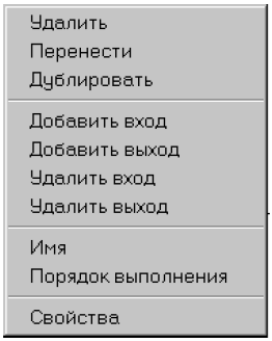


Количество циклических выходов может зависеть от количества входов (тогда они называются циклическими, зависящими от входов), если они являются циклическими (как в примере), тогда при изменении количества входов, меняется и количество выходов, и наоборот.

2. Нециклические – выходы, количество которых не может быть изменено и задается автоматически при создании блока.

Добавление/удаление входов и выходов.

Для того, чтобы добавить / удалить вход или выход функционального блока щелкните правой кнопкой мыши на функциональном блоке, в результате чего появится меню, в котором следует выбрать соответ-

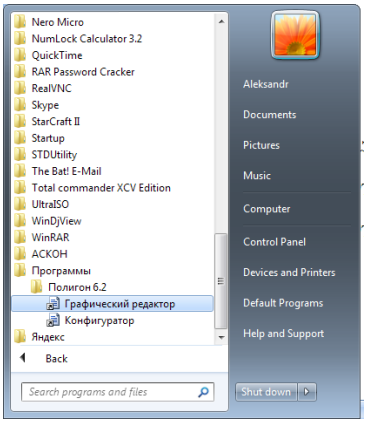


ствующую команду. При выполнении данных операций все связи, на неизмененных входах и выходах сохраняются. При удалении входа или выхода, на которые были проведены связи, эти связи также удаляются.

Если данные операции производятся с функциональным блоком, у которого количество выходов зависит от количества входов (см. раздел Выходы), то все действия производятся относительно входов и приводят к одновременному изменению количества входов и выходов.

1.3. ПОРЯДОК ВЫПОЛНЕНИЯ

1. Внимательно изучить общие указания



2. Открыть программу **Конфигуратор** из папки **Полигон**.

Окно конфигуратора представлено на рис. 1.4

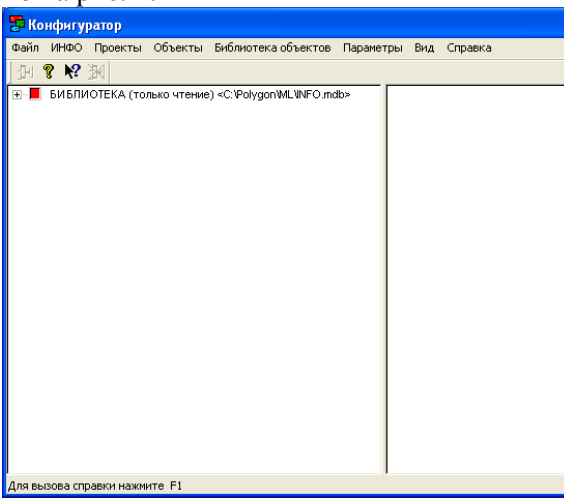
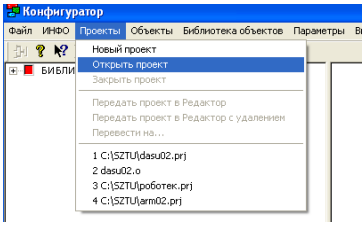


Рис. 1.4.

3. Создать проект в **Конфигураторе (Проекты→Новый проект).**



Вновь созданный проект разместить в папке SZTU на диске C (C:\SZTU). Имя проекта должно иметь формат, допустимый для имени файла (не содержать символов русского алфавита, символов ! " « » № ; % : ? * ( ) -=+)

После этого программа примет следующий вид (см. рис. 1.5.)

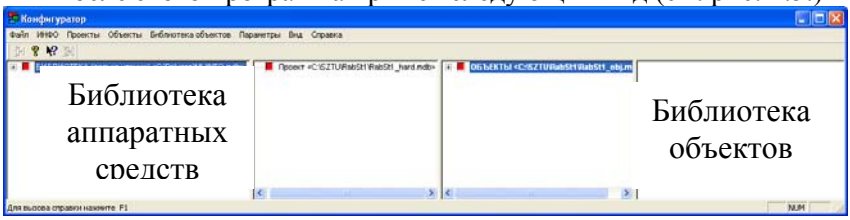


Рис. 1.5

4. Из окна **Библиотеки аппаратных средств** перенести необходимые модули КОНТРОЛЛЕРА в окно **Аппаратная часть проекта** в соответствии с рис 1.6. для имеющейся конфигурации PC-ПК Beckhoff CX1000-00.

Полная конфигурация проекта представлена в приложении 1.

Сначала переносится из меню **Терминалы** базовый элемент промышленного контроллер **CX1000+1100Kbus.** Он содержит процессорный блок и интерфейсы. Модули ввода–вывода подключаются к **Kbus** шине.

При переносе модулей ввода-вывода из меню **Модули** (рис. 1.6.) сначала переносятся модуле дискретного вывода **KL2xy8 interface Kbus** (в количестве 3 шт.) и присоединяются к шине **Kbus** терминального модуля **CX1000+1100 interface Kbus**, затем переносятся модули ввода **KL1xy8 interface Kbus**. Расположение блоков в дереве должно соответствовать расположению в контроллере.

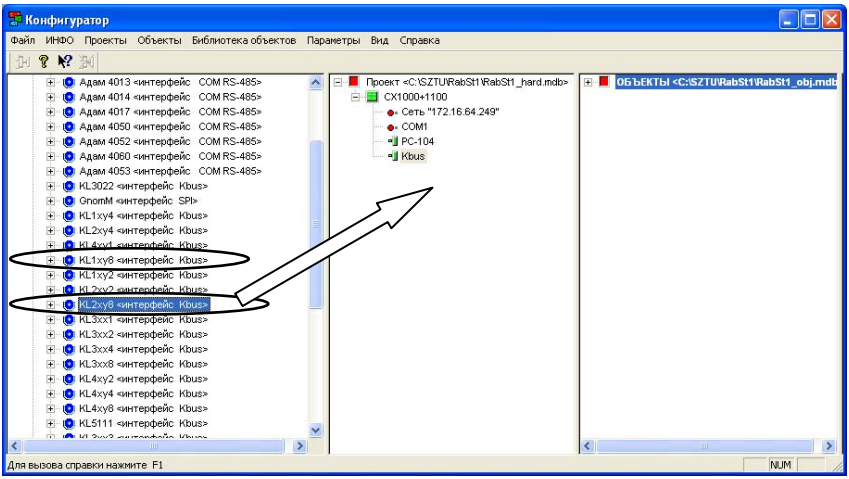
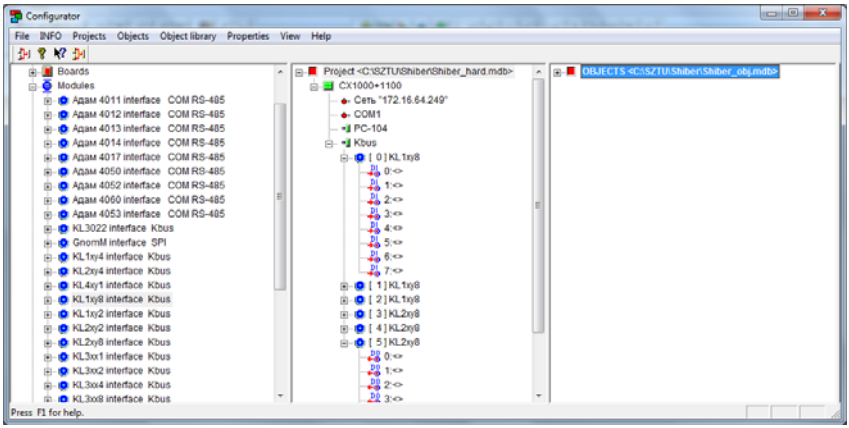
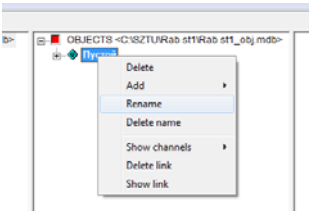


Рис. 1.6.

После переноса аппаратная часть выглядит следующим образом:



5. Далее необходимо описать структуру объекта управления в окне **Объекты.** Для переименования объекта используется контекстное меню.
Переименуем его в **Раб станция**



6. Создадим узел **Станок**.
   Добавление узлов производится через контекстное меню **Добавить → Узел.**
7. В созданных узлах сформируем канал **M4 C1 мотор.** Конфигурирование канала и присваивания ему имени производится в меню, представленном на рис. 1.7.

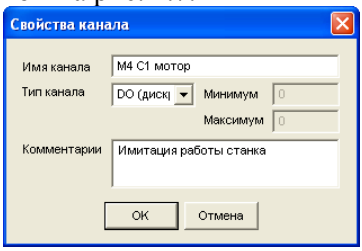


Рис. 1.7.

Рабочее поле объекты должно принять вид, показанный на рис. 1.8.

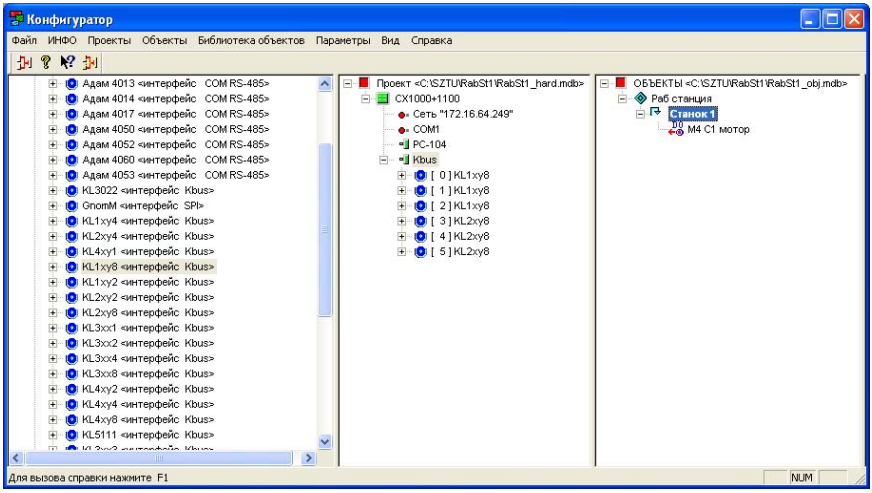


Рис. 1.8.

8. Установить связи между аппаратной частью и объектом. Связи образуются перетаскиванием каналов объекта к каналам аппа-

ратной части. При перетаскивании необходимо подтверждать установление связей (см. рис. 1.9).

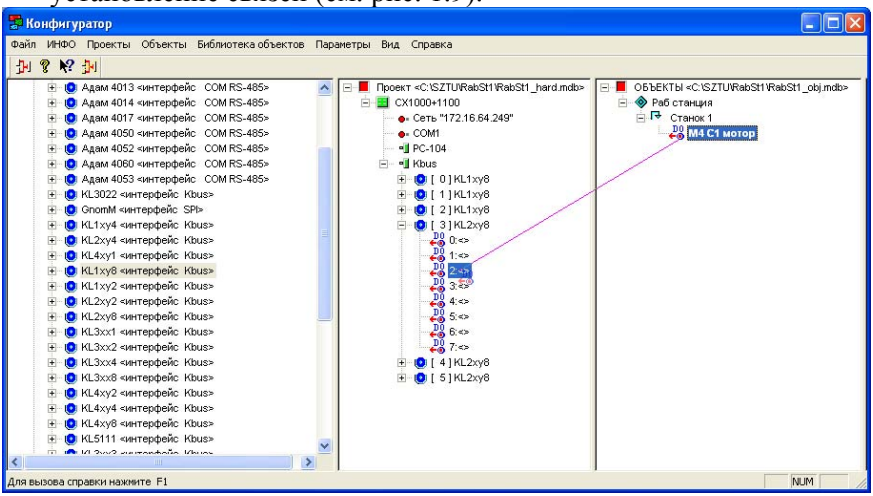
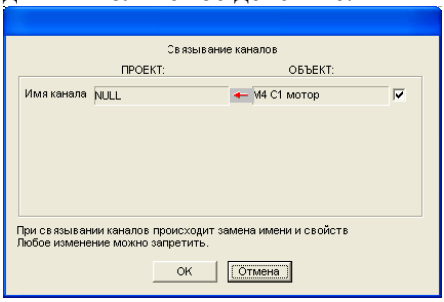
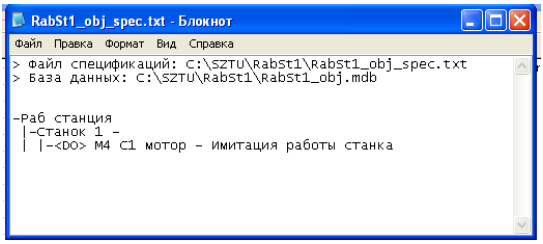


Рис. 1.9

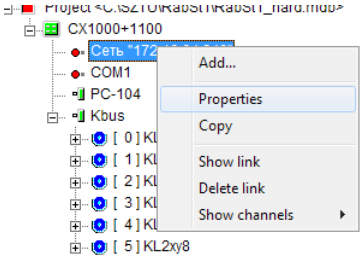
Подтверидим выполняемое действие.



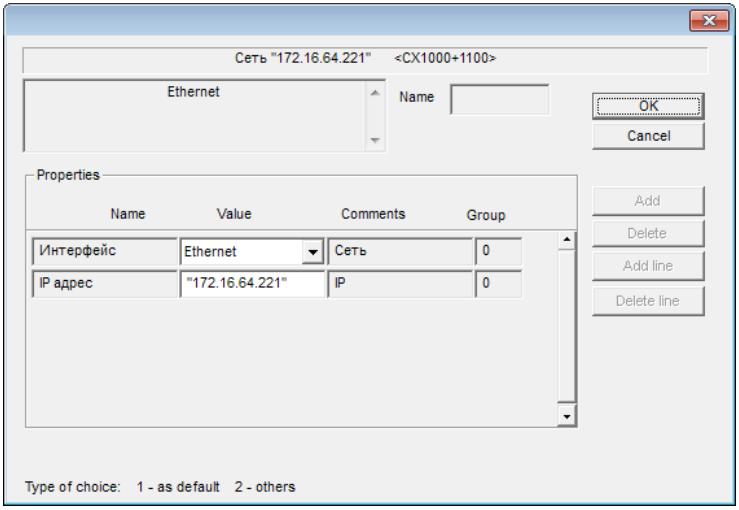
Дерево объектов можно вывести в текстовый документ формата **.txt**. Для этого надо щелкнуть на нем левой кнопкой мыши и выбрать команду **Создать файл** спецификации в меню **Объекты**. При этом будет создан текстовый файл следующего вида.



9. Установим IP адрес контроллера. Для этого, выделив **Сеть,** в контекстном меню выберем раздел **Свойства**.



В появившимся диалоговом окне установим следующий IP адрес контроллера. IP адрес контроллера стенда – 172.16.64.221



10. После конфигурирования аппаратной части ее необходимо передать в Графический редактор. Для этого в меню **Проект** выбирает пункт **Передать в Графическкий редактор** (рис. 1.8)**.** На во-

прос о необходимости удаления комментариев отвечаем нет (рис. 1.10).

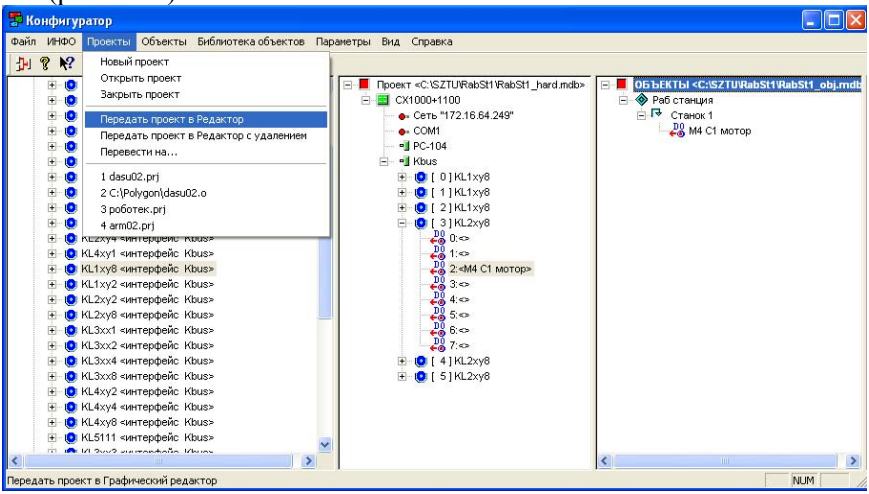


Рис. 1.10

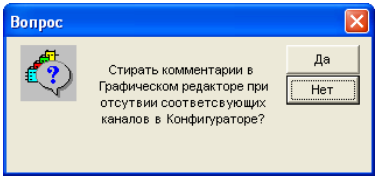
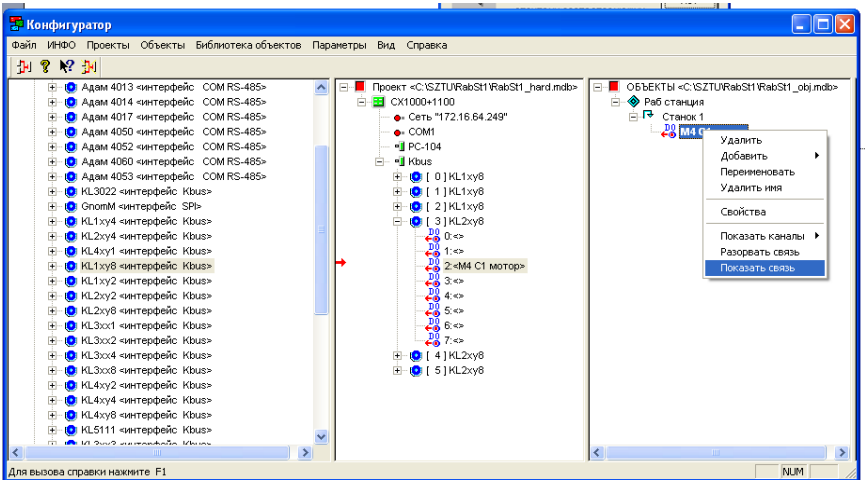


Рис. 1.11

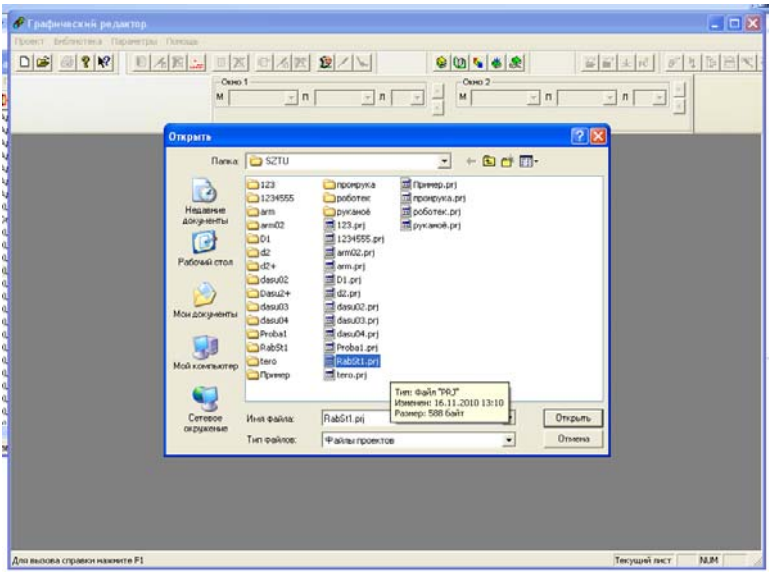
11. Посмотреть связи можно через контекстное меню



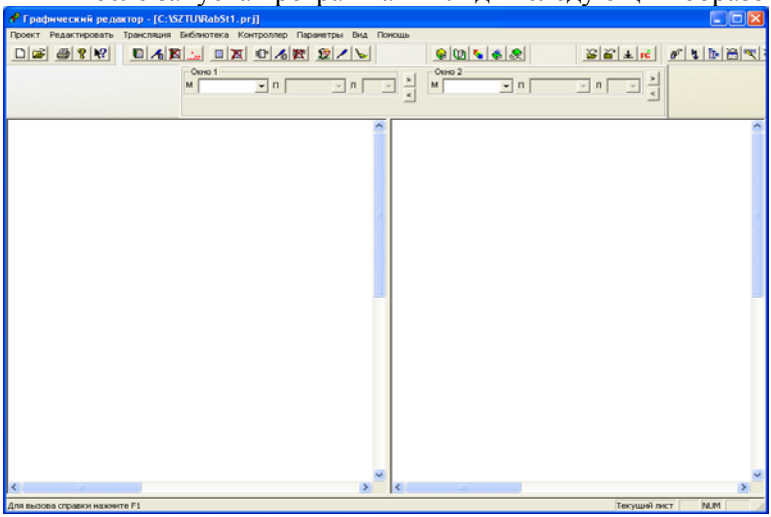
На этом описание аппаратной части СУ и его согласование с ОУ, и соответственно работа в **Конфигураторе** заканчивается и можно перейти в **Графическкий редактор** для создания собственно управляющей программы.

РАБОТА В ГРАФИЧЕСКОМ РЕДАКТОРЕ

12. Открыть Графический редактор и через меню Проекты открыть ранее созданный проект (файл с расширением *.prj).



После запуска программа выглядит следующим образом:



13. Созданные в конфигураторе выходной канал можно найти во **Входы → Inpprg** (см. рис. 1.12) на листе 2 выходы (ставшие в программе входами данных) и на листе 3 входы.

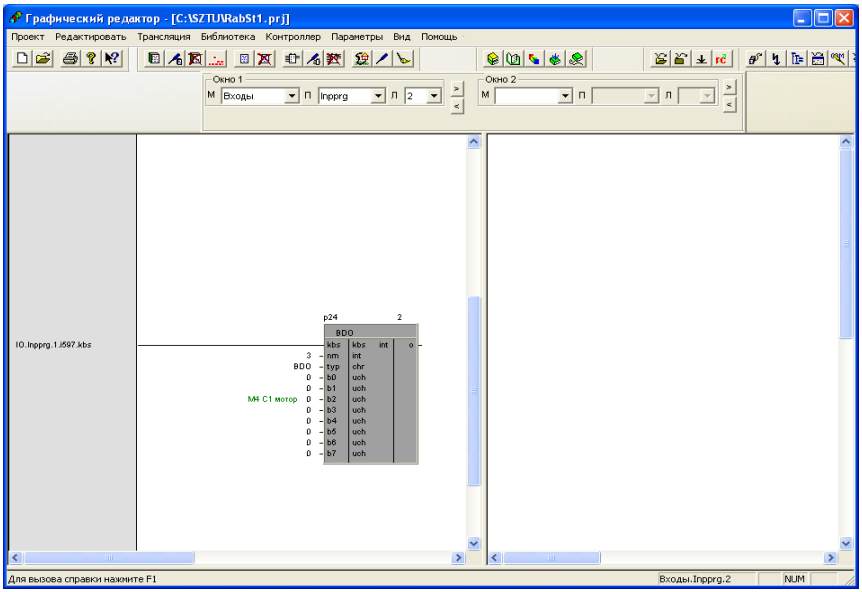
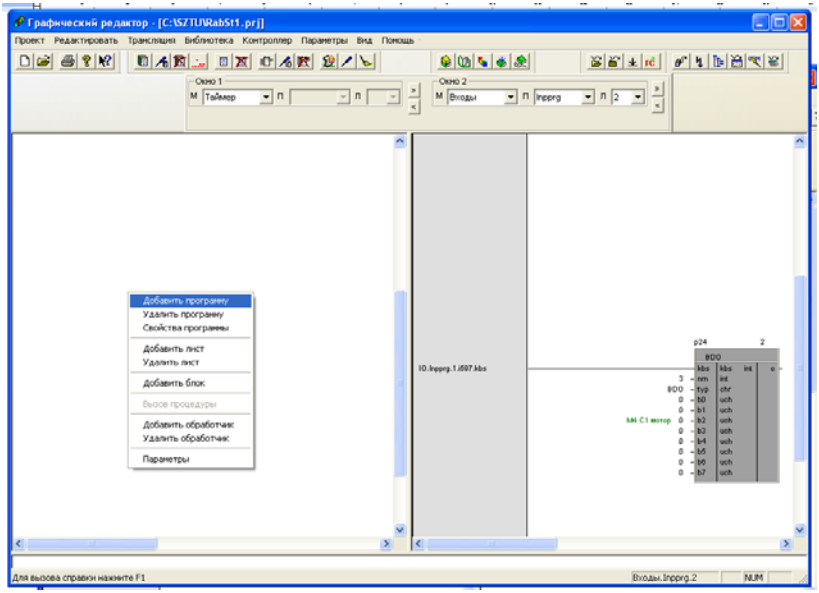


Рис. 1.12

14. Для описания алгоритма управления необходимо добавить ПРОГРАММУ. Для того, чтобы создать программу:
    1. Выберите в меню **Редактировать → Программа** команду **Создать** или нажмите кнопку  на Панели инструментов;



2. В появившемся блоке диалога задайте параметры программы. В последствии эти параметры можно изменить (команда Параметры в меню Редактировать→Программа), для этого надо сделать активным какой-нибудь лист программы (рис.1.13).

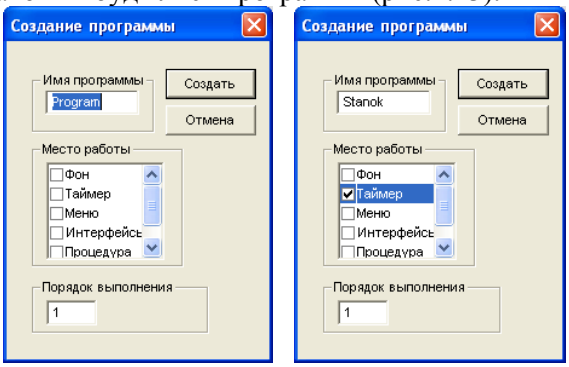
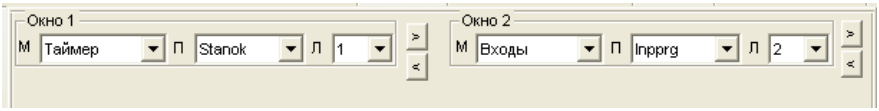


Рис. 1.13. Диалоговое окно **Создание программы** в момент вызова и после заполнения соответственно

В панели выбора листа (место **Таймер**) появится программа **Stanok**



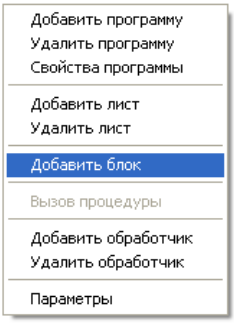
15. Управляющая программа представляет собой совокупность функциональных блоков, преобразующих информацию со входов в выходные сигналы в соответствии с заданным алгоритмом.

Чтобы создать функциональный блок выполните одно из следующих действий:

1. Выберите команду **Создать** в меню **Редактировать→Блок**;

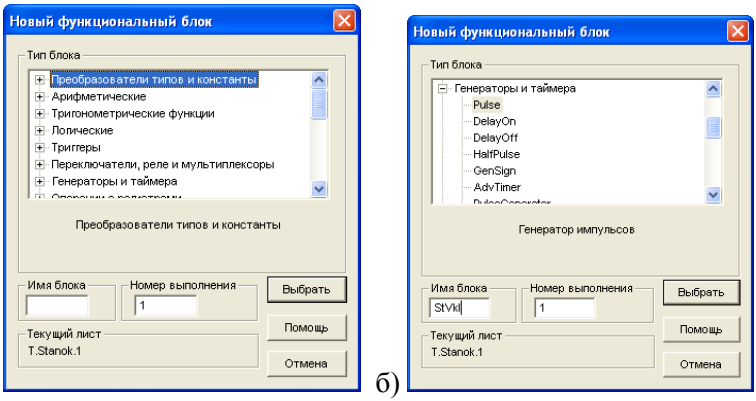
2. Нажмите кнопку  на Панели инструментов;

3. Нажмите **Alt+Б**.



После этого появится блок диалога (рис. 1.14а), в котором задаются параметры функционального блока. Для получения более подробной информации о конкретном типе функционального блока необходимо выбрать этот тип в окне **Тип блока** и нажать кнопку **Помощь** *(в текущей версии не работает)*.

Если входы или выходы данного блока являются циклическими (см. раздел **Входы** п. Общие указания), то предварительно выводится блок диалога, в котором задается их количество. В окне необходимо задать имя блока и порядок его выполнения на листе (рис. 1.14б).

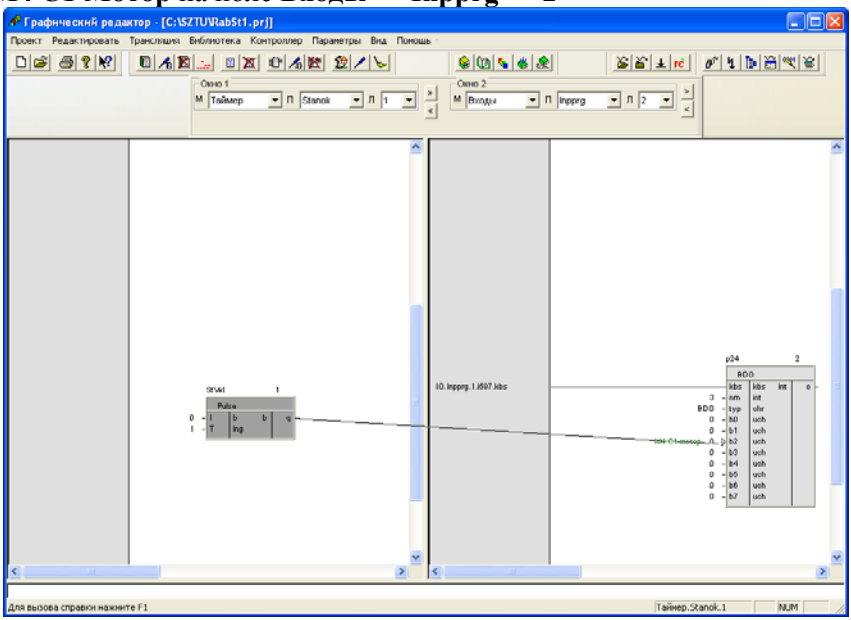


а) б)

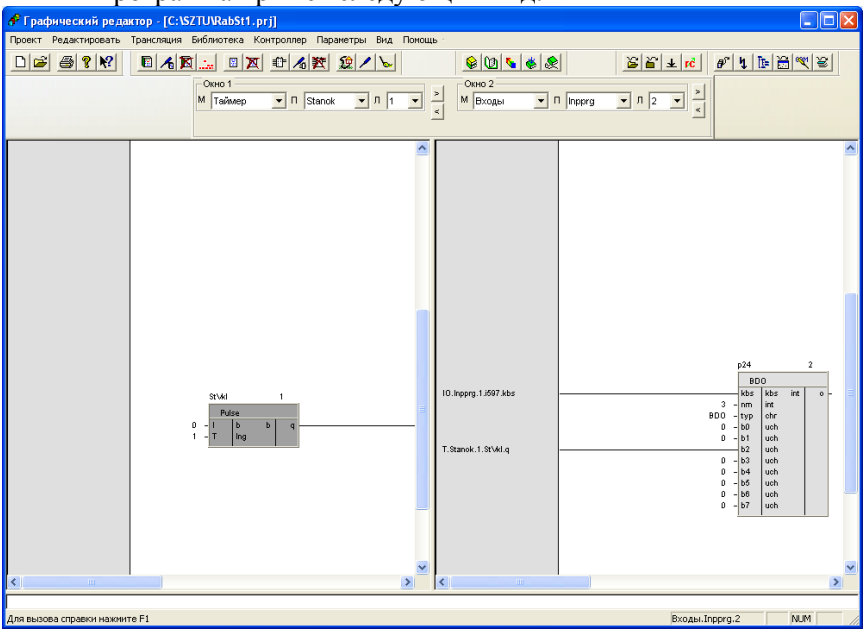
Рис. 1.14.

После нажатия кнопки **Выбрать** на **Текущем листе** будет создан функциональный блок с заданными параметрами.

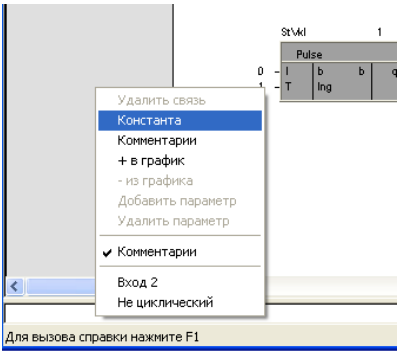
16. Соединим выход созданного блока с входом датчика **M4 C1 Мотор** на поле **Входы → Inpprg → 2**



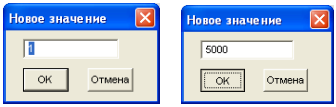
Программа примет следующий вид:



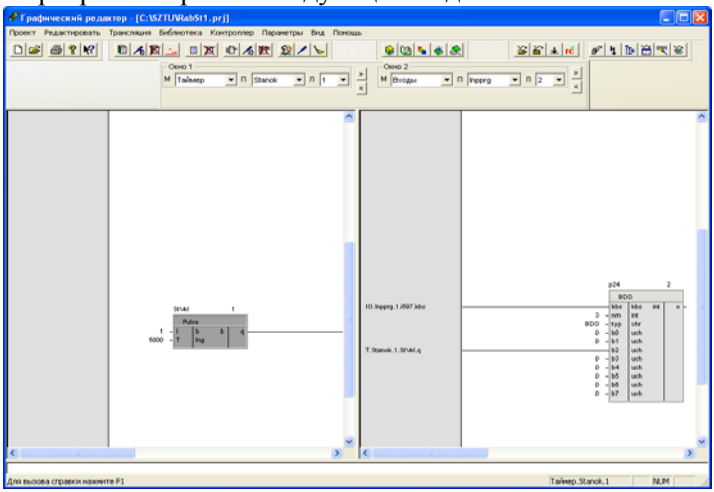
17. Определим время работы станка задав **Константу**, используя контекстное меню.



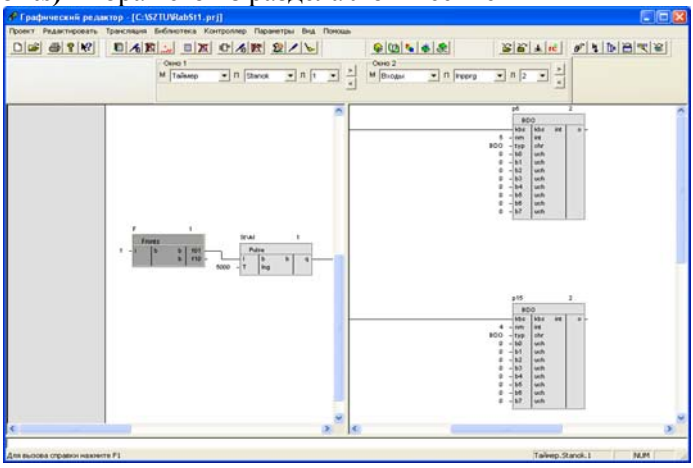
Согласно заданию установим время задержки в 5 мс



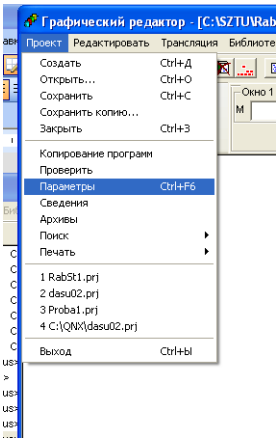
Программа примет следующий вид:



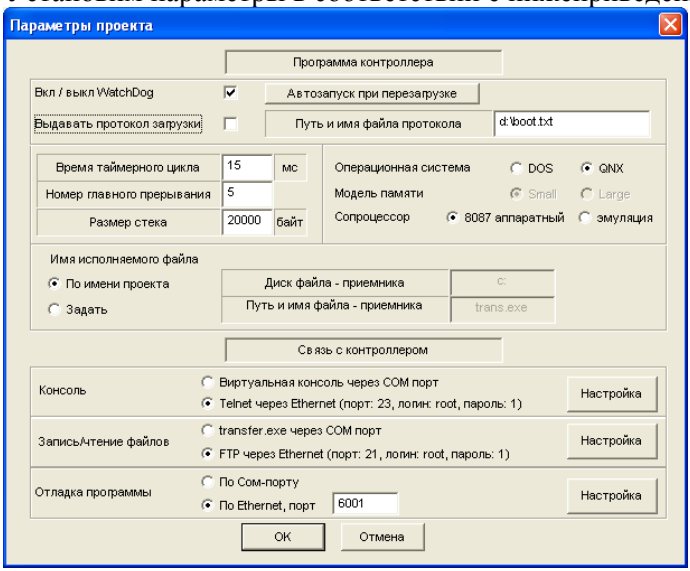
18. Аналогичным образом создадим еще блок **Детектор фронтов (Fronts)** выбрав его из раздела Логические



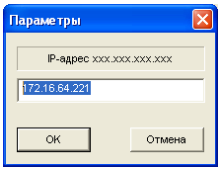
19. Зададим параметры проекта. Для этого войдем в меню **Проект → Параметры**.



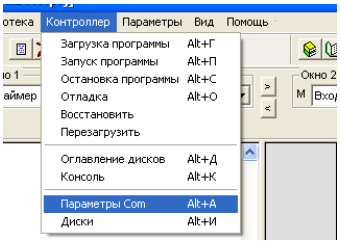
Установим параметры в соответствии с нижеприведенным:

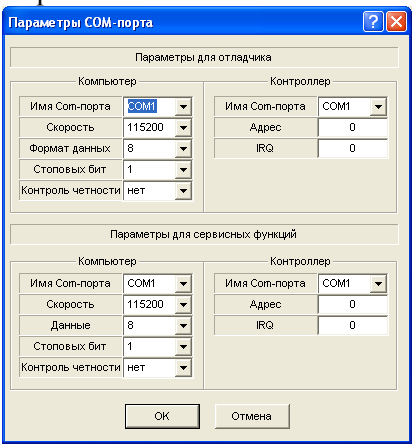


Нажав на любую из трех кнопок **Настройка** введем IP адрес 172.16.64.221.

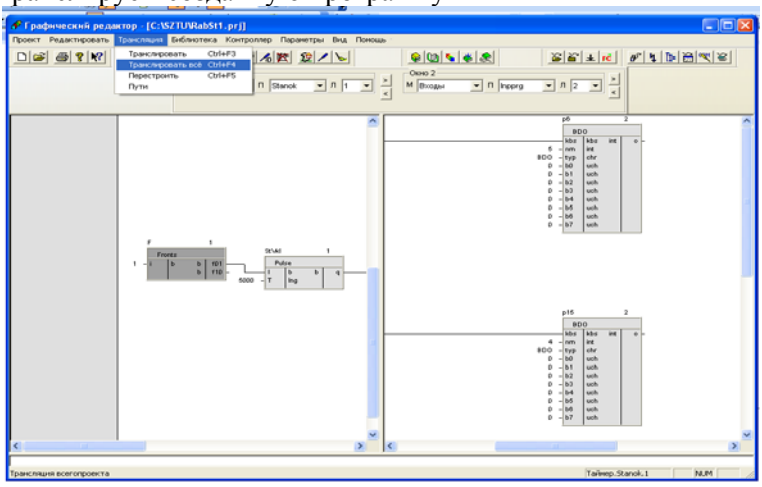


Введем параметры COM порта

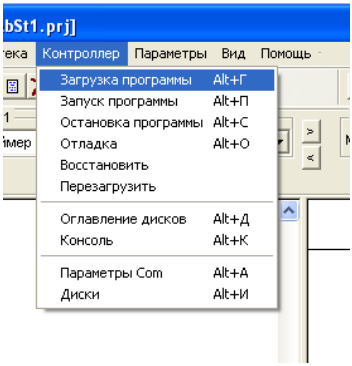


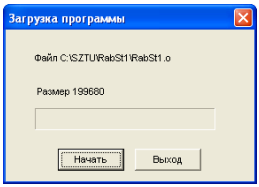


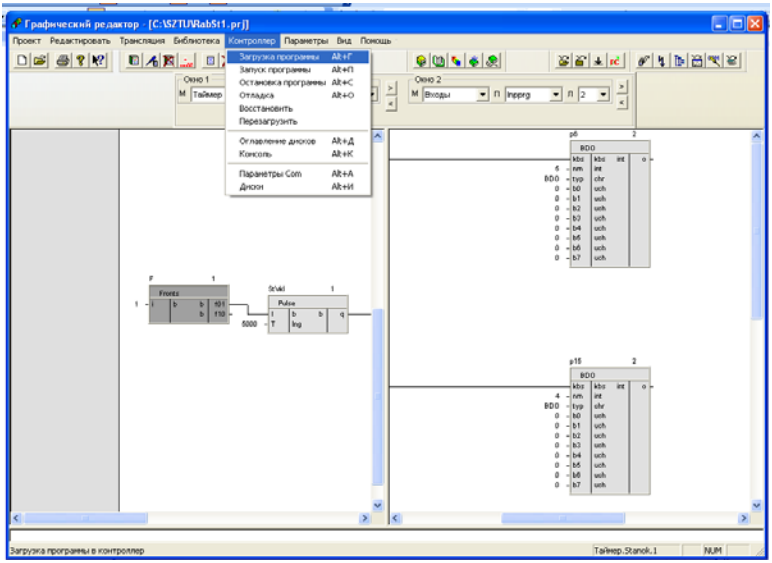
20. Транслируем созданную программу



21. Запуск и отладка программы производятся из меню **Контроллер** или кнопок на панели управления  . Даная работа проводится ТОЛЬКО в присутствии преподавателя.







Лабораторная работа №2

РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ ПОТОЧНОЙ ЛИНИИ

1.1 ЦЕЛЬ РАБОТЫ

Получение навыков программирования промышленных контроллеров для решения задач управлением вспомогательным оборудованием в машиностроительной отрасли.

1.2 ЗАДАНИЕ

Разработать программу управления конвейером рабочей станции рис. 2.1. для контроллера Beckhoff CX1000-000

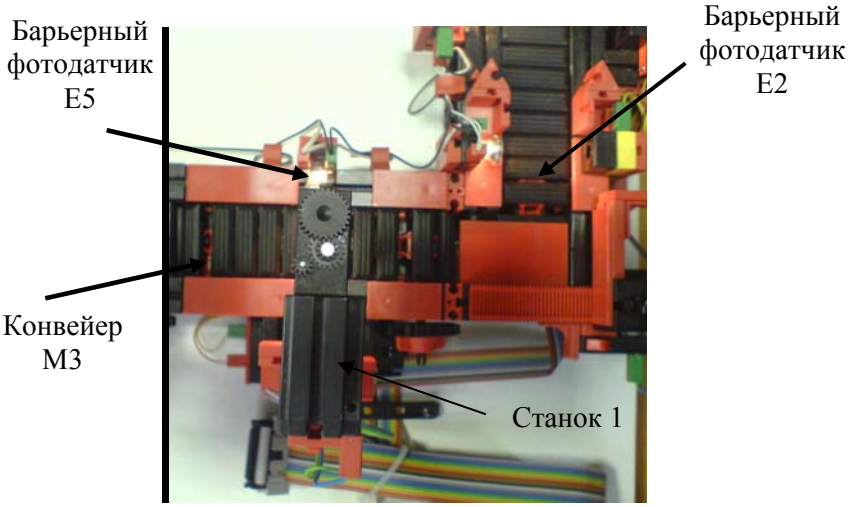


Рис. 2.1.

Конвейер служит для перемежения детали в рабочую зону имитатора станка. Срабатывание фотодатчика **E5 готовность станок 1** означает, что деталь подана на конвейер станка 1 и он может приступить к ее обработке.

Запуск программы осуществляется по сигналу от оптического датчика **E2 загружено** и спустя две секунды запускается мотор

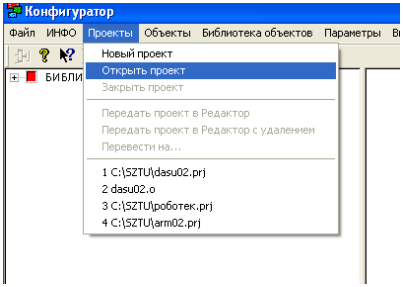
конвейера **M3 C1 транспортер**. По достижении **E5 готовность станок 1** транспортер останавливается и включается двигатель станка **M4 C1 станок** на 5 секунд. По истечении заданного времени запускается мотор конвейера **M3 C1 транспортер** на 3 секунды.

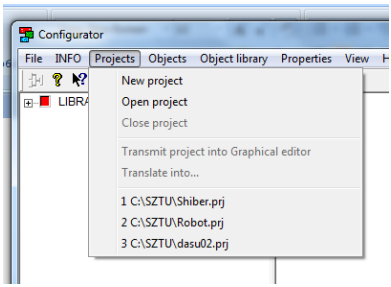
2.2. ПОРЯДОК ВЫПОЛНЕНИЯ

1. Изучить алгоритм управления.
2. Открыть конфигуратор

РАБОТА В КОНФИГУРАТОРЕ

3. Создать новый проект в **Конфигураторе (Проект→Создать).**





4. Перенести (см. рис. 2.2) необходимые модули КОНТРОЛЛЕРА из окна **Библиотеки** (разделы Терминалы и Модули) в окно **Аппаратная часть проекта** в соответствии с рис 2.3. для имеющейся конфигурации PC-ПК Beckhoff CX1000-000
   При переносе **Модулей** сначала переносятся модуле дискретного вывода **KL2xy8 interface Kbus** (в количестве 3 шт.) и присоединяются к шине **Kbus** терминального модуля **CX1000+1100 interface Kbus**, затем переносятся модули ввода **KL1xy8 interface Kbus**.

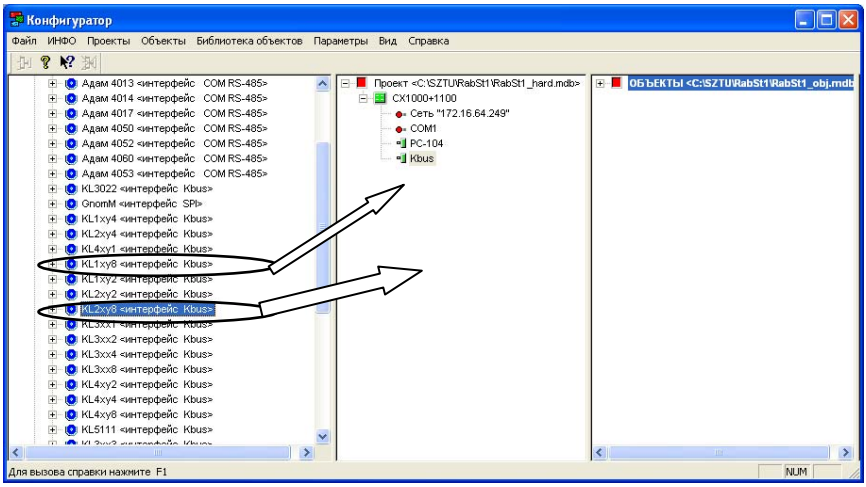


Рис. 2.2.

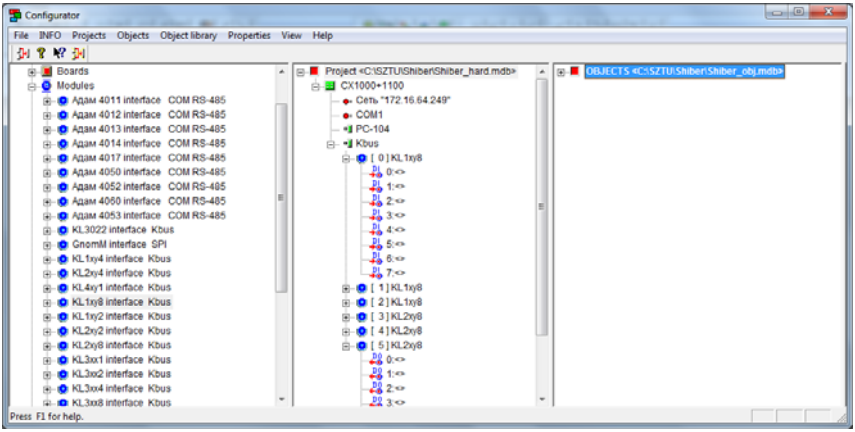


Рис. 2.3.

5. Элементы объекта управления можно ввести вручную или скопировать из библиотеки 4 окно (рис. 2.4), открыв из меню **Библиотека объектов** диалоговое окно **Открыть** (рис. 2.5) и выбрав в папке **dasu02** файл **dasu02_obj.**

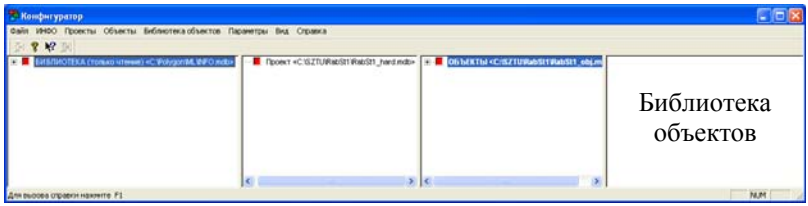


Рис. 2.4.

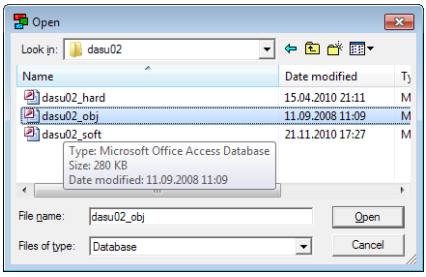


Рис. 2.5.

Элементы дерева объектов можно копировать, а также можно открыть в соседнем окне библиотеку объектов (команда **Открыть** меню **Библиотека объектов**), из которой копировать элементы. Библиотекой объектов может служить, например, база данных объектов другого проекта. Полностью проект «Макет поточной линии» в **Конфигураторе** представлен в приложении 1 (файл **dasu02.prj**).

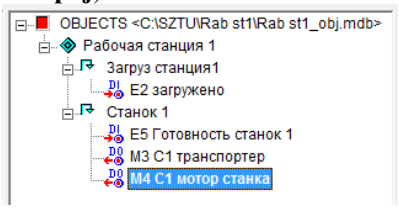


Рис. 2.6

Описать структуру объекта управления в окне **Объекты** как показано для рассматриваемого примера на рис. 2.6.

Для переименования объекта используется контекстное меню (рис. 2.7). Создадим два узла **Загрузочная станция 1** и **Станок 1.** Добавление узлов производится через контекстное меню **Добавить → Узел.**

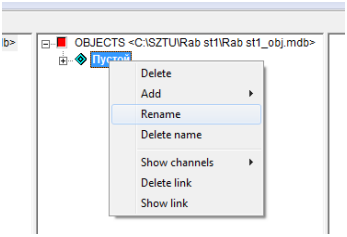


Рис. 2.7.

В созданных узлах сформируем каналы. Запуск конвейера осуществляет дискретный вход **E2 загружено** в узле **Загрузочная станция 1.**

Добавление каналов производится через контекстное меню **Добавить → Канал.** (рис. 2.8). Конфигурирование канала и присваивания ему имени производится в меню, представленном на рис. 2.9.

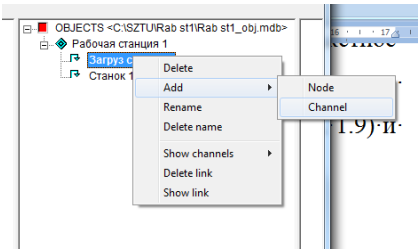


Рис. 2.8.

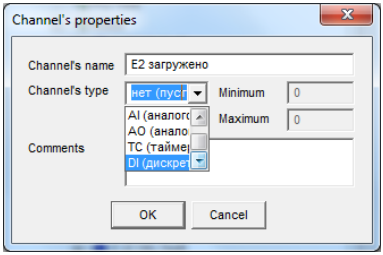


Рис. 2.9.

Для управления конвейером необходимо добавить еще один дискретный канала ввода и два вывода (см. рис. 2.6).

6. Установить связи между аппаратной частью и объектом. Связи образуются перетаскиванием каналов объекта к каналам аппаратной части. При перетаскивании необходимо подтверждать установление связей (см. рис. 2.10).

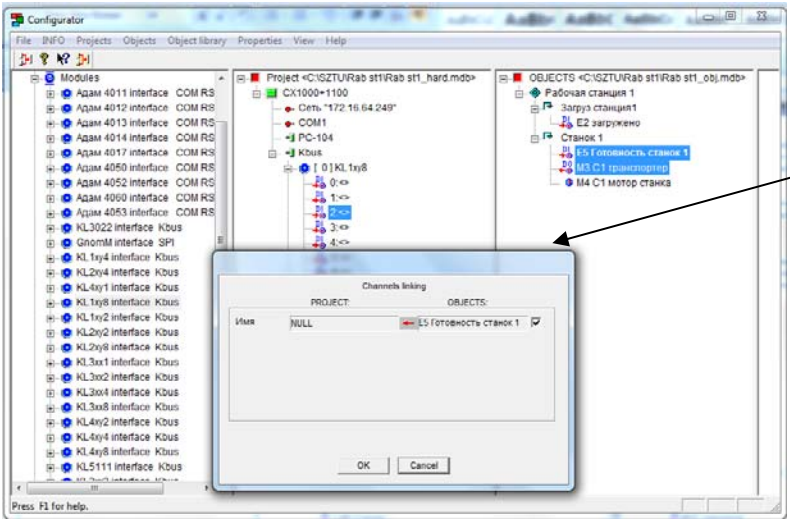


Рис. 2.10.

Привязка объекта к аппаратной части представлена на рис. 2.11.

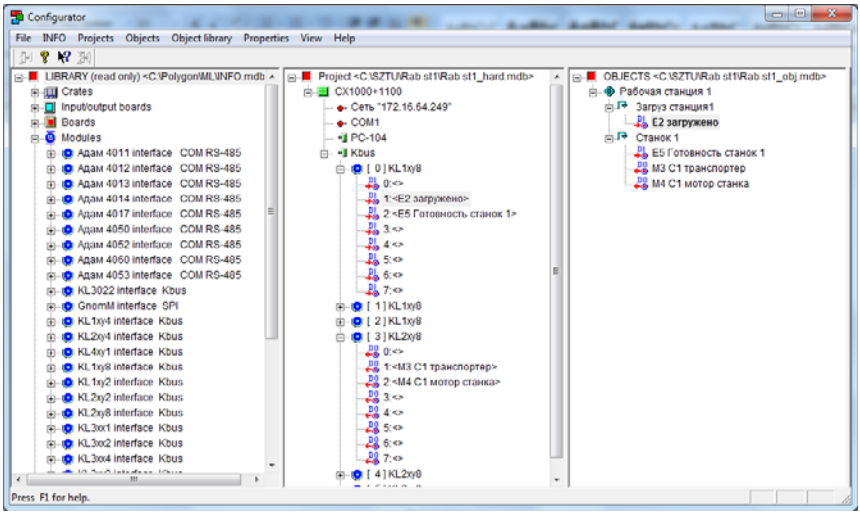


Рис. 2.11.

7. После конфигурирования аппаратной части ее необходимо передать в Графический редактор. Для этого в меню **Проект** выбирает пункт **Передать в Графическкий редактор.** На вопрос о необходимости удаления комментариев отвечаем нет (рис. 2.12).

Рис. 2.12.

На этом работа в **Конфигураторе** заканчивается и можно перейти в **Графическкий редактор** для создания собственно управляющей программы.

РАБОТА В ГРАФИЧЕСКОМ РЕДАКТОРЕ

8. Открыть **Графический редактор** и через меню **Проекты** открыть ранее созданный проект (файл с расширением *.prj).
9. Созданные в конфигураторе входы и выходы можно найти во **Входы → Inpprg** (см. рис. 2.13) на листе 2 выходы (ставшие в программе входами данных) и на листе 3 входы.

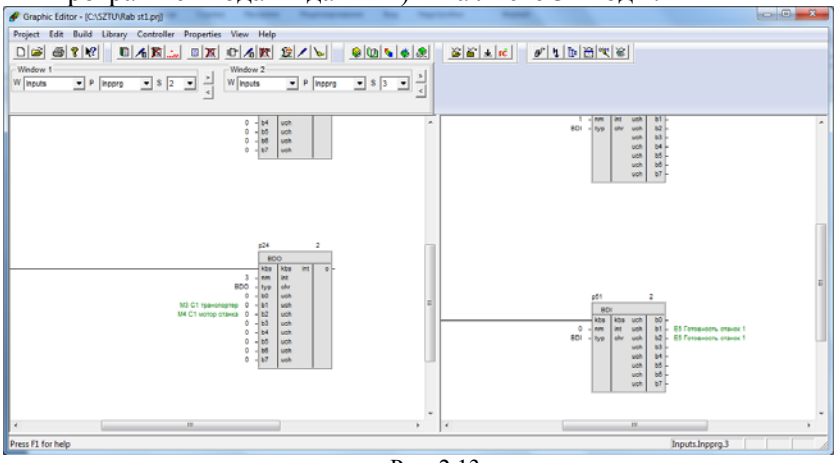
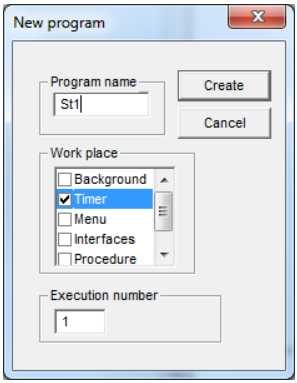


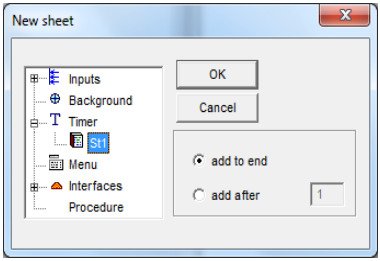
Рис. 2.13.

10. Для написания программы управления конвейером необходимо добавить ПРОГРАММУ. Для того, чтобы создать программу:

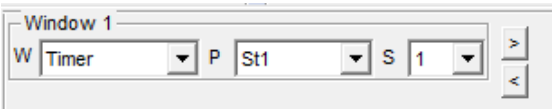


1. Выберите в меню **Редактировать → Программа** команду **Создать** или нажмите кнопку  на Панели инструментов;

2. В появившемся блоке диалога задайте параметры программы. В последствии эти параметры можно изменить (команда **Параметры** в меню **Редактировать → Программа**), для этого надо сделать активным какой-нибудь лист программы.



В панели выбора листа (место **Таймер**) появится программа **St1**



11. Добавим лист. Для этого вызовем контекстное меню и выберем **Добавить лист**

12. Выбрав второй лист программы **St1** создадим управляющую программу, которая представляет собой совокупность функциональных блоков, преобразующих информацию со входов в выходные сигналы в соответствии с заданным алгоритмом.

Чтобы создать функциональный блок выполните одно из следующих действий:

1. Выберите команду **Создать** в меню **Редактировать→Блок**;

2. Нажмите кнопку  на Панели инструментов;

3. Нажмите **Alt+Б**.

После этого появится блок диалога, в котором задаются параметры функционального блока. Для получения более подробной информации о конкретном типе функционального блока необходимо выбрать этот тип в окне **Тип блока** и нажать кнопку **Помощь**.

Если входы или выходы данного блока являются циклическими (см. раздел **Входы** п. Общие указания), то предварительно выводится блок диалога, в котором задается их количество.

После нажатия кнопки **Выбрать** на **Текущем листе** будет создан функциональный блок с заданными параметрами.

13. Инвертируем сигнал с датчика **E2 Загружено**, являющегося нормально замкнутым логическим элементом. Для этого разместим на поле слева **Входы → Inpprg → 3,** справа откроем второй лист программы **St1.** В правом окне создадим **Блок** вызвав контекстное меню и выбрав **Добавить блок**

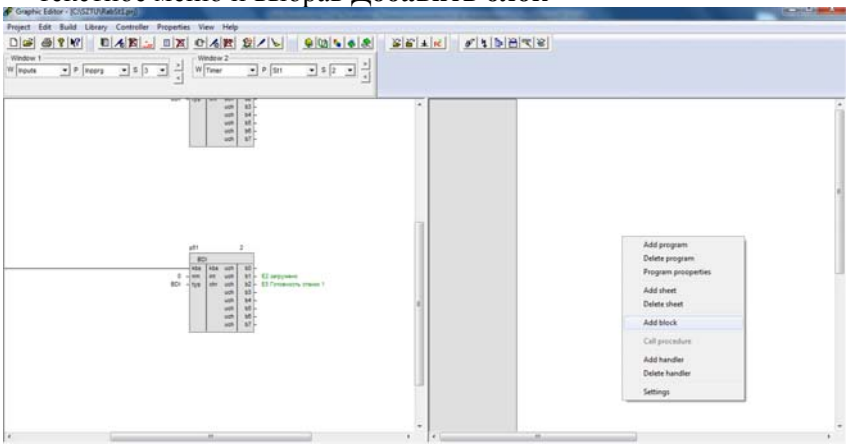


Рис. 2.14.

В появившемся меню выберем **Логические → NOT**. Зададим имя **InvE2** блоки и порядок выполнения (**1**) согласно рис. 2.15

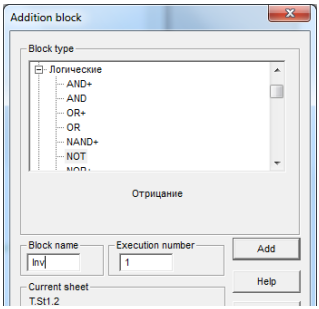


Рис. 2.15.

Соединим вход созданного блока с выходом датчика **E2 загружено** на поле **Входы → Inpprg → 3** (рис**.** 2.16.)

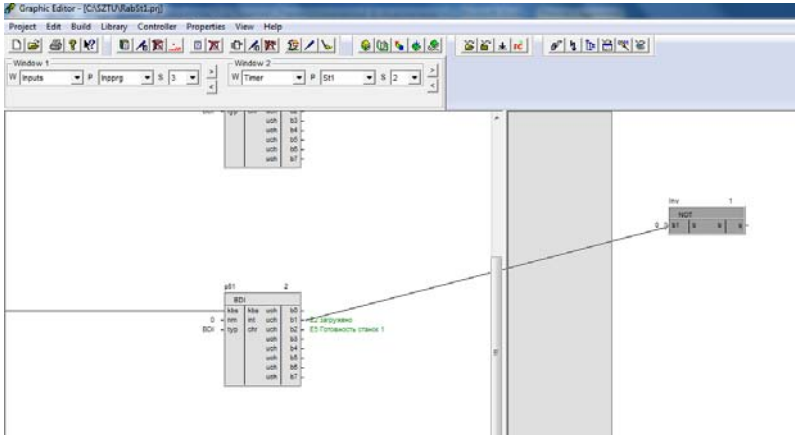


Рис. 2.16.

14. Добавим блоки **Детектор фронтов** из раздела **Логические** и **RS-триггер с R-доминантой с инициализацией выхода(RSR_i)** из раздела **Триггеры.** Соединим элементы как показано на схеме рис. 2.17

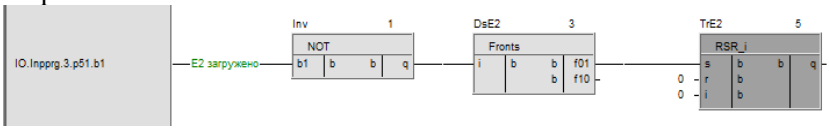


Рис. 2.17.

15. При срабатывании датчика **E5 Готовность станок 1** необходимо остановить конвейер. Это условие подадим на вход триггера

TrE2. Для этого добавим блоки **Отрицание(NOT)** и **Детектор фронтов(Fronts)**, подключив их к входу E5 **Готовность станок 1** аналогично **E2 загружено** (рис. 2.18).

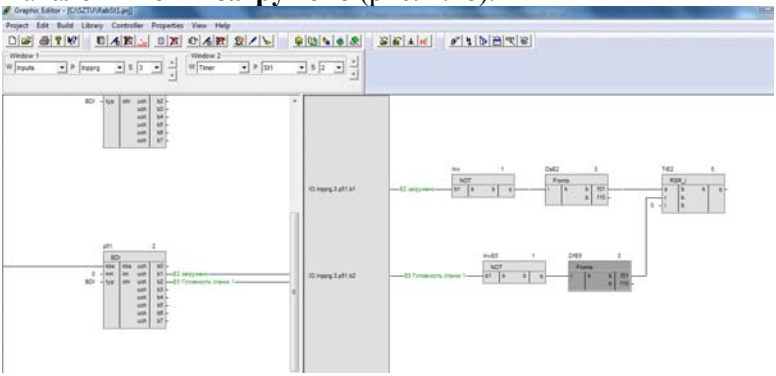
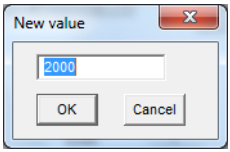


Рис. 2.18.

16. Для формирования двухсекундной задержки на включение установим блок **Включение с задержкой** из раздела **Генераторы и таймеры**. На входе блока **T** установим время задержки. Установив курсор на входе, вызовем контекстное меню и выберем пункт **Константа**. В появившемся меню установим время задержки включения 2000 мс.



17. Для документирования программы добавим комментарий (рис. 2.19), выбрав соответствующий пункт из контекстного меню.

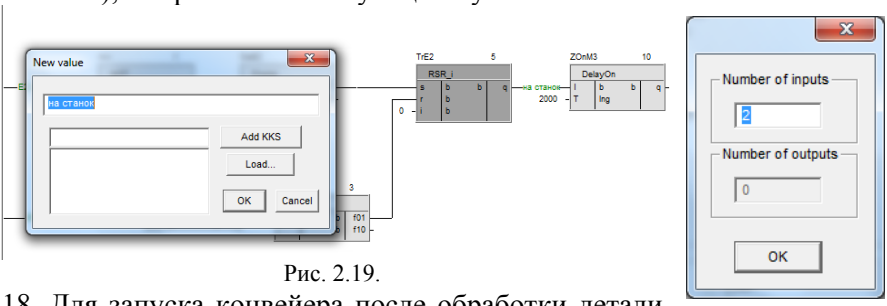


Рис. 2.19.

18. Для запуска конвейера после обработки детали (второе условие начала работы) добавим блок **ИЛИ**. После зада-

ния имени блока и порядка выполнения появится диалоговое окно задания количества входов. Оставим два по умолчанию.

19. Расположив рабочее окно программы **St1** слева и открыв на правой половине рабочего окна **Входы → Inpprg → 2**, соединить выход блока или со входом М3 С1 транспортер. Общий вид программного кода показан на рис. 2.20.

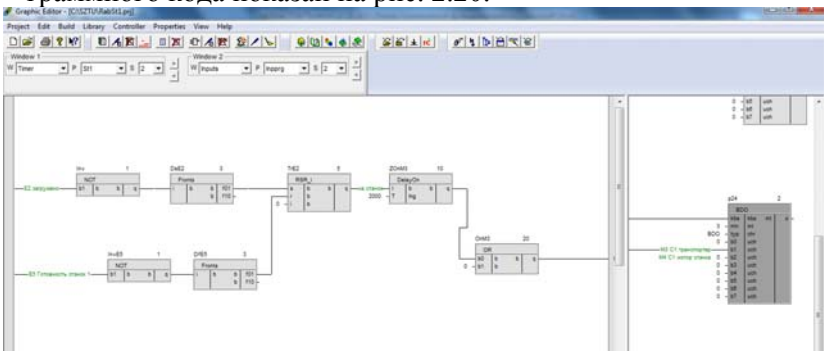


Рис. 2.20.

20. По аналогии реализовать всю управляющую программу. Для удобства работы рекомендуется создать новый лист в программе для формирования кода управления мотором станка.
21. В качестве примера можно использовать программный код исходного проекта (файл **dasu02.prj**)

Лабораторная работа №3

РАЗРАБОТКА ПРОГРАММЫ УПРАВЛЕНИЯ ШИБЕРОМ ЗАГРУЗОЧНОЙ СТАНЦИИ

1.2 Цель работы

Получение навыков программирования промышленных контроллеров для решения задач рефлекторного управления совместной работы оборудования в машиностроительной отрасли.

1.3 ЗАДАНИЕ

Разработать программу управления шибером загрузочной станции рис. 3.1. для контроллера Beckhoff CX1000-000

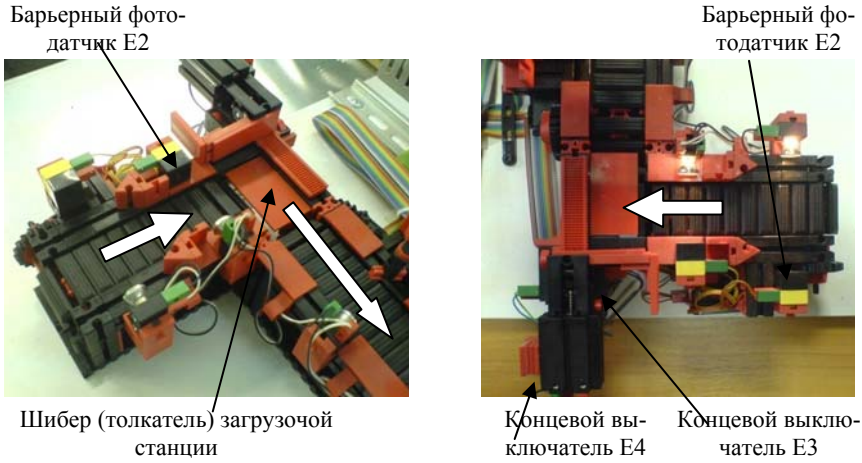


Рис. 3.1.

Шибер служит для перемежения детали поданной с загрузочного конвейера на конвейер станка. Крайние положения толкателя контролируется двумя концевиками (**E3 ЗС шибер концевик вперед** и **E4 ЗС шибер концевик назад**). Срабатывание концевого выключателя **E3 ЗС шибер концевик вперед** означает, что деталь подана на конвейер станка 1 и он может приступить к ее обработке.

После запуска программы необходимо удостоверится, что шибер находится в крайнем положении (т.е. выключатель **E3** находится в замкнутом положении). В случае его разомкнутого состояния запустить двигатель шибера **M1 назад** для возврата его в исходное положение.

Запуск шибера **M1 вперед** осуществляется по сигналу от оптического датчика **E2** после выдержки времени в одну секунду. При достижении конечного выключателя **E3 ЗС шибер концевик вперед** двигатель шибера переключается на **M1 назад** для возврата его в исходное положение. Во время выполнения рабочего цикла сигналы с оптического датчика **E2** игнорируются.

1.4 ПОРЯДОК ВЫПОЛНЕНИЯ

1. Изучить алгоритм управления.
2. Перенести (см. Л.Р. 1, рис. 1.3) необходимые модули из окна **Библиотеки** (разделы Терминалы и Модули) в окно **Аппаратная часть проекта** в соответствии с рис 1.5. для имеющейся конфигурации PC-ПК Beckhoff CX1000-000

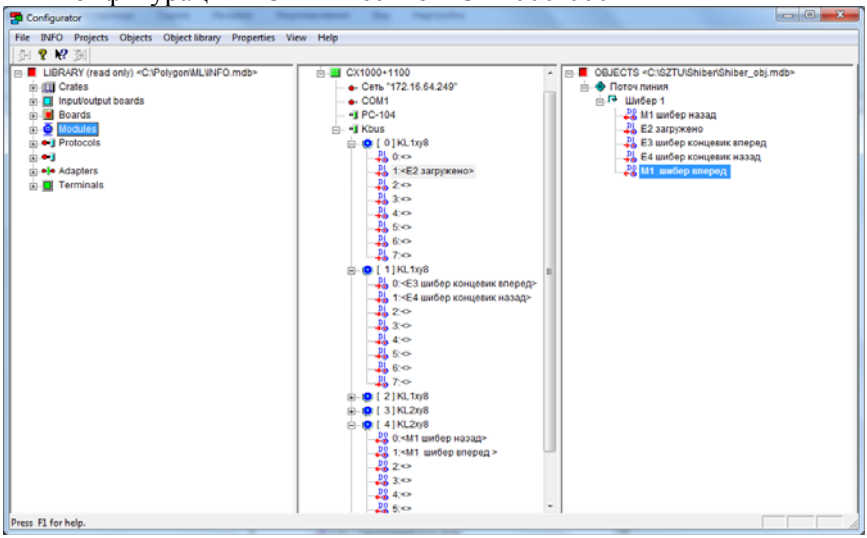


Рис. 3.2.

3. В качестве примера можно использовать программный код исходного проекта (файл **dasu02.prj**)

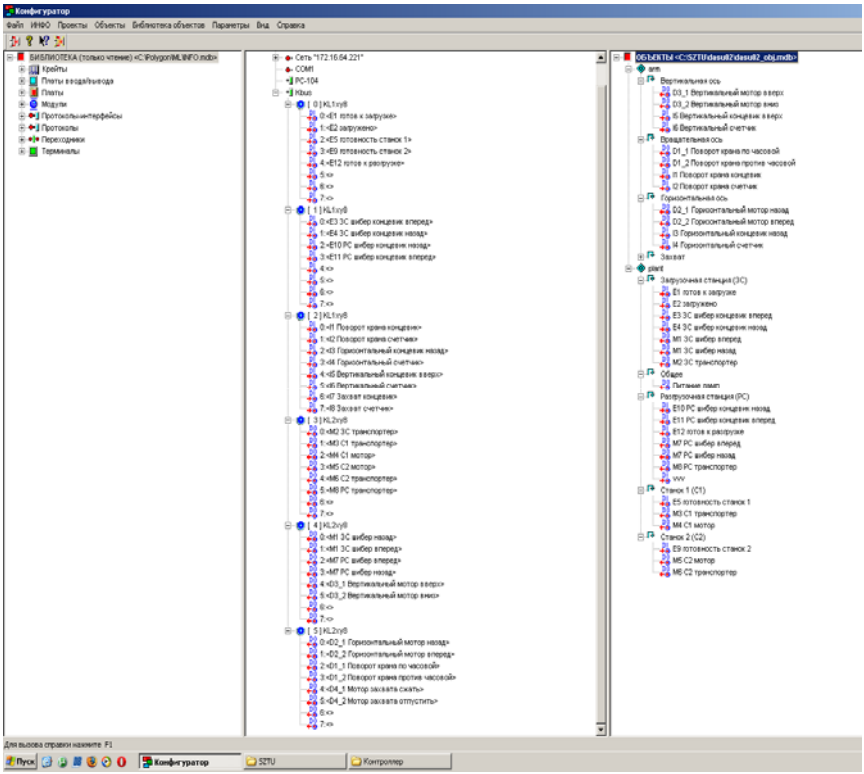
БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Автоматизация производственных процессов в машиностроении: учеб. пособие / Е.Э. Фельдштейн, М.А. Корниевич. - Минск: Новое знание; М.: ИНФРА-М, 2017, http://znanium.com/bookread2.php?book=884475
2. *Кибанов, А. Я.* Автоматизация управления машиностроительным предприятием [Электронный ресурс]: Учебное пособие для слушателей заочных курсов повышения квалификации ИТР по организации управления машиностроительным производством / А. *Я. Кибанов, Т. А. Родкина*. М.: Машиностроение, 1989 http://znanium.com/bookread2.php?book=432601
3. Система разработки программного обеспечения для ПК-совместимых контроллеров Полигон. Руководство пользователя – СПб.: Промавтоматика, 2015

Содержание

Демонстрационный стенд ДАСУ-02 6
Лабораторная работа №1. Разработка программного обеспечения контроллеров в среде «полигон» 10
Лабораторная работа №2. Разработка программы управления оборудованием поточной линии 39
Лабораторная работа №3. Разработка программы управления шибером загрузочной станции 51

Приложение 1



СПЕЦИАЛЬНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ В МАШИНОСТРОЕНИИ

ПРОГРАММИРОВАНИЕ ПОТОЧНЫХ ЛИНИЙ

Методические указания к лабораторным работам для студентов магистратуры направления 15.04.04

Сост. *А.А. Кульчицкий*

Печатается с оригинал-макета, подготовленного кафедрой автоматизации технологических процессов и производств

Ответственный за выпуск *А.А. Кульчицкий*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 24.01.2020. Формат 60×84/16.
Усл. печ. л. 3,1. Усл.кр.-отт. 3,1. Уч.-изд.л. 2,7. Тираж 50 экз. Заказ 29. С 2.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2